# User's Manual

NPRDC Tactical Training System
Version 5.2
October 17, 1982

## Forward (12/20/2008)

Looking back 25 years to one of the earliest projects you worked on is an interesting task. I think this manual and the below references are all that I know that was published on this interesting system. I was one of the programmers, and also had a hand in writing the manual. Other folks associated with this project included Jim Hollan, Ed Hutchins, Steve Putz, Tim Mcandless, Mark Rosenstein (me) and Sean Malloy. I'm sure there were others as I came onto the project well after it started. I can still hear the noise of the impact dot matrix printer that generated this paper.

Mark.

## References

Crawford, A. M. and Hollan, J. D. (1983). Development of a computer-based tactical training system. (NPRDC-SR-83-13). San Diego, CA: Navy Personnel Research and Development Center, San Diego, Calif., January 1983.

Halff, Henry M.; Hollan, James D.; Hutchins, Edwin L. Cognitive science and military training. American Psychologist. Vol 41(10), Oct 1986, 1131-1139.

## Introduction

This manual describes the NPRDC Tactical Training System, Version 5.2. The system provides a learning environment using computer games that test and drill the user on information contained in a database. The student encounters two parts of the system : the games and a database. The games depend only on the structure of the database, not the subject matter represented in the database. This makes the games useful to a broad range of users.

Contained in this document are detailed descriptions intended for three possibly overlapping audiences. For the student, there is a chapter on using the games. For the instructor, there is a lengthy discussion of reducing the subject matter of interest into a database. This discussion includes descriptions of the support programs that make this process less painful. The last member of the audience is the system administrator. This person must have at least a rudimentary understanding of the UCSD Pascal System. This will be the person who maintains the disks, doing various necessary utility functions.

The present system is implemented on two types of hardware as detailed below:

```
     TERAK 8510 with 8515 top drive

     Apple II+ with either 2 floppy disks
                    or a winchester drive
               with Videx VideoTerm Card
               with Language Card
```

The System is delivered on floppy disks. Below you will find a listing of the files that should be present on the disks for each system:

```
     TERAK: (three 8 inch floppy diskettes)
          Sys_5.2 Disk Contains:
             SYSTEM.8510.QB
             SYSTEM.8510.QX
             SYSTEM.PASCAL
             SYSTEM.STARTUP
             FILER.CODE
             SYSHELP.CODE
             CLEANAUDIT.CODE
             CONAUDIT.CODE
             SYSTEM.CHARSET
             PART.CODE
             TWENTY.CODE
             PICQUIZ.CODE
             SHOW.CODE
             JEOPARDY.CODE
             CONCEN.CODE
             CONSTRAINT.CODE
             FLASH.CODE
             GAME.INDEX.TEXT
```

```
Student Disk Contains:
    GAME.COM
    CATEGORIES.HELP
    STARTUP.HELP
    CONCEN.HELP
    SPACE.INDEX
    SPACE.NAMES
    SPACE.NODES
    PARTS.INDEX
    SHOW.1/2CHARSET
    PARTS.NAMES
    COMPON.TEXT
    SWAPFILE
    AUDIT.TEXT
    AUDIT.FILL
    PARTS.NODES
    SYSTEM.HELP
    PIX.INDEX
    PIX.PICS
    CITIES.INDEX
    CITIES.NODES
    CITIES.NAMES
    COUNTRIES.NODES
    COUNTRIES.NAMES
    COUNTRIES.INDEX
    DATABASES.TEXT
    PART.HELP
    INDEX.MX
    MATFILE.MX
    FLASH.HELP
    SHOW.HELP
    KEYS.HELP
    CONSTRAINT.HELP
    TWENTY.HELP
    MATRIX.HELP
    UC_SOVIET.INDEX
    UC_SOVIET.NAMES
    UC_SOVIET.NODES

Administration Disk Contains:
    SYSTEM.8510.OB
    SYSTEM.8510.QX
    SYSTEM.CHARSET
    SYSTEM.PASCAL
    SETUP.CODE
    MAKEREC.CODE
    SETNAME.CODE
    SYSHELP.CODE
    CONAUDIT.CODE
    CLEANAUDIT.CODE
    FILER.CODE
    SYSTEM.STARTUP
    EDITOR.CODE
    BROWSE.CODE
    LEARN.CODE
    DUMPER.CODE
```

UTIL.INDEX.TEXT
DBEDIT.CODE

Apple:

The next chapter of this manual will discuss getting started with the system. The third chapter has a subsection for each game and will describe in detail how the games should be played. The fourth chapter describes making and editing databases. The fifth chapter describes other utilities that are supplied with the gaming system.

A final note for this introduction is in order. This manual should not be taken too seriously. Nothing here is hard. Perhaps, things are poorly explained, but certainly not hard. If following the instructions in the next section, copies are made of the system disks and experimentation is made using these copies, just playing around with the system can do no damage, and experimentation is the way to become familar with the gaming system. Few lives have been lost playing these games and reports flow in of people actually managing to use this system to learn things. (Imagine that!!!).

## What to Do as Soon as the Original Disks are Received

Terak Version

For the Terak version of the games, a modified keyboard is used. This means that some of the keys have different meanings to the Gaming System than that marked on the keys. The location and proper labeling of the keys is given in Appendix?. What we have found to be convient is to take mailing labels, cut small squares the size of the modified keys, write the appropriate new function name on the label, cover the label with scotch tape (so the ink doesn't smear under constant use), and place the labels on the keys. For instance, over the { key, we place a label: SELECT.

The first thing to do, when the nice new System floppies are received is to make copies of them, so if worse comes to worst only a copy will be destroyed. The following procedure is the only time where not following instructions could cause 'bad things' to happen.

First, turn the Terak on. Turn on both of the switches on the upper right hand front corner of the 8510 and the 8515. You should hear the fans come on in each of the units. If not, check that they are plugged in. If this doesn't work check the Terak owners manual.

When the Terak is on, insert the floppy disk marked ADMINISTRATION in the 8510. The disk is inserted label up with the edge not having the label on it inserted first. Close the disk drive door. After the disk has been read, the prompt at the top of the Terak screen will appear:

```
          Welcome to the NPRDC Tactical Gaming System

          Please place a database disk in the top drive.
          Press RETURN when ready.
```

At this point press the ESC key (or the QUIT key if you have relabeled the keyboard), which is the upper leftmost key on the keyboard. The system now responds with a screen:

```
NPRDC Tactical Gaming System[Administration Utilities]



   Select one of the following:
 ________________________________________________________
> DBedit - Database Editor                               |
| Setup - Create a Student Disk                          |
|             .                                          |
|             .                                          |
|             .                                          |
| Filer - File Utilities                                 |
| Editor - Text File Editor                              |
|________________________________________________________|

ARROWS move cursor, SELECT to select a utility, QUIT if done, or HELP
```

Notice that in front of DBedit there is a >. This indicates the computer is now pointing to DBedit. The > is moved using the up and down arrow keys located to the left of the numeric keyboard which is located on the right side of the keyboard.

Use the down arrow key to move the > in front of the line:

```
        , Filer - File Utilities
```

If the > is accidentally moved too far use the up arrow key to move it back on the Filer line.

Now press the "<" key which if the above instructions for labeling the keys have been done correctly, should be labeled: SELECT.

When this is done, the line is highlighted(dark characters on a light backround) for a second, then the screen clears and at the top of the screen is:

```
        Filer:G(et,S(ave,W(hat,N(ew....etc
```

COPYING THE ADMIN DISK

To copy the ADMIN disk, put a new disk in the 8515 drive, label up, label edge last in as in fig.1. There shouldn't be anything on this disk, if there is it will get destroyed. After closing the door, press T.

Now the screen clears and you will see the following question:

```
     Transfer what file?
```

You should type in the following:  #4,#5

then press RETURN

The next line of text on the screen appears right below the preovious line and it says:

```
     Transfer 494 blocks? (Y/N)
```

You should type: Y

Another line of text appears directly below the previous line and says:

```
      Destroy (name of your disk)?
```

You should type: Y

Now all the files will be copied on to your disk from the ADMIN disk. When the coping is complete you will see this text:

```
   ADM5.2                    ---> (name of your disk)
```

## COPYING THE SYSTEM DISK

Take both disk out of the drives by pushing in on the 2 inch by 3/8 inch rectangle that has a light in the middle of it. It is positioned right below the door . Replace these disks as follows:

a. Put the SYSTEM disk in drive 8510 and shut the door by pushihg it down.

b. To copy the SYSTEM disk, put a new disk in the 8515 drive, label up, label edge last in as in fig.1. There shouldn't be anything on this disk, if there is it will get destroyed. After closing the door, press T.

Now the screen clears and you will see the following question:

```
Transfer what file?
```

You should type in the following: #4,#5

then press RETURN

The next line of text on the screen appears right below the preovious line and it says:

```
Transfer 494 blocks? (Y/N)
```

You should type: Y

Another line of text appears directly below the previous line and says:

```
Destroy (name of your disk)?
```

You should type: Y

Now all the files will be copied on to your disk from the SYSTEM disk. When the coping is complete you will see this text:

```
SYSTEM                              ---> (name of your disk)
```

## COPYING THE STUDENT DISK

Now once more take both disks out of the drives by pushing in on the 2 inch by 3/8 inch rectangle. Replace these disks as follows:

a. Put the STUDENT disk in drive 8510 and shut the door by pushihg it down.

b. To copy the STUDENT disk, put a new disk in the 8515 drive, label up, label edge last in as in fig.1. There shouldn't be anything on this disk, if there is it will get destroyed. After closing the door, press T.

Now the screen clears and you will see the following question:

```
        Transfer what file?
```

You should type in the following:  #4,#5

then press RETURN

The next line of text on the screen appears right below the preovious line and it says:

```
       Transfer 494 blocks? (Y/N)
```

You should type: Y

Another line of text appears directly below the previous line and says:

```
         Destroy (name of your disk)?
```

You should type: Y

Now all the files will be copied on to your disk from the STUDENT disk. When the coping is complete you will see this text:

```
        STUDENT                             ---> (name of your disk)
```

## Getting Started

### Terak Version

Take a disk labeled Sys_5.2 and a student disk, possibly with your name on it. Turn the computer on, by turning on the button on the top right side of the front of both the 8510 on the 8515. Put the Sys 5.2 disk in the 8515 drive and the student disk in the 8510 drive. The proper orientation of the disk is shown in fig. 1. The label should be pointing upwards, and the non label edge should be put in first. Next, close the doors. If everything has been done correctly (wasn't too tough) the system should respond,

```
    Welcome to the NPRDC Tactical Gaming System, version 5.2

   Is today 23-Jun-82? [YES/NO]
```

Systems are A Ok now!!!

If instead the following message appears on the screen,

```
     Welcome to the NPRDC Tactical Gaming System, version 5.2

   Please place a database disk in the top drive.
   Press RETURN when ready.
```

then check to make sure the student disk (the one in the top drive) is inserted correctly, checking the orientation against fig. 1. Also check to make sure the door has been closed securely. After completing these checks, press the RETURN key, which is labeled not suprisingly, return, and is located next to the row of keys which have arrows on them.

If the message appears again ask a system administrator for help. Similarly, if no message appears, check your steps again and press the on button on the 8515 after the check has been completed. If still nothing happens, ask a system administrator.

If all has gone correctly, the system is now waiting for the answer to the question:

```
             Is today's date, June 23, 1982.
```

The part of the system that asks ~~the~~ initial questions is called the gameshell (answering the famous question, which came first the chicken or the egg. Answer: the shell). The [YES/NO] after the question indicates that the computer expects either the YES or the NO key to be pressed. In Appendix ? the layout for the special keys is shown. If the keyboard you are using hasn't been modified, please refer to the appendix to find the location of the special keys mentioned here and elsewhere in this document. If the date has changed(very likely) just press the NO button. If this is done, the system responds:

```
     Enter today's date: [day-month-year]
```

It is only necessary to type in the first three letters of the months name. So if for instance you are playing the games at your New Year Eve's Party (before midnight while you were still sober) you would type:

    31-DEC-82

You may have noticed that DEC came out in capital letters. To get upper and lower letters press the DC2 key (it is located in the numeric pad on the right side of the keyboard(it is also labeled the 2 key). After the 82 in the date you will have to press the RETURN key. In almost all cases after you type in more than a single letter you will have to press the RETURN key. This key tells the computer that you are done typing. After RETURN is pressed the computer responds:

    Is today 31-Dec-82 [YES/NO]

(if you typed in another date it will of course have that date). If this date is correct press the YES key. If you have typed wrong press the NO key and try it again. After the date has been set, the system responds:

    Are you familiar with the Gaming System[YES/NO]?

This line introduces another fine characteristic of the Gaming System called On-Line-Help (soon you too can speak computerese: gameshell On-Line-Help, your vocabulary is growing apace). What On-Line-Help provides is a quick explaination or reminder of what to do. As you sit frozen in fear all you need do is press the HELP key, located second from the top in the leftmost row of keys on the keyboard. This will cause a short explaination to appear. Also, when a new phase of the system is entered, such as presently when the gameshell begins, an opportunity to receive help will appear.

The question:

    Are you familiar with the Gaming System?

if answered NO will provide a short description of the game system. If you are an old pro with the current part of the system just press the YES key (meaning I'm too smart for the computer's silly explaination) and no explaination will be generated. If this is the first time through, press the NO button. The system will begin a quick explaination of the gaming system.

A typical Help will provide a number of lines of explanation. At the bottom of the explaination will be:

    Press NEXT to continue, BACK to return...

This line is called the prompt line. A prompt line can appear either at the top or bottom of the screen. This line indicates which commands will be accepted by the system. At this point, the appropriate commands are NEXT and BACK. If the NEXT button is pressed, another page of help will be placed on the screen. If the

BACK button is pressed, then the help will terminate and the system will continue from the point where HELP was pressed or after NO was pressed in response to the question:

Are you familar with ...

After the help, the main part of the gameshell begins. It is here that the game to be played is chosen. The main gameshell screen is shown in Fig. 2. The organization of the screen as shown there is typical of many places in the Gaming System. At the top, there is the header information telling what program is presently operating. The present heading:

Welcome to the NPRDC Tactical Gaming System[5.2]
Hello, Sir

indicates the gameshell is operating, and if the system administrator has been at work, the Sir in this example should be replaced with your name.

At the bottom, there is the familar prompt line telling which commands the system will understand. There are some new options, most of which are used with the mysterious structure in the middle of the screen.

This structure is called a menu. As in the menu from a restaurant, it is possible to select items from this menu. In this case, it is possible to select a game to play. On the keyboard, notice the vertical column of keys, which are labeled with arrows ( <-- , --> , ^, and down arrow). These keys are used to move around in a menu or between menus. Since there is one menu, only the up and down arrows will be used at present. The menu contains:

Concentration (with apologies to Hugh Downs )
Constraint - the game of narrowing choices
Flash - an interesting way to spend some time
Parts Quiz - the visual stimuli response game
Picture Quiz - the game of pattern recognition
Show - a graphical means of information retrieval
Twenty Questions - a game of knowlege
Jeopardy - The Answer Game

These are the games of the Gaming System. In front of Concentration, there is a >. Try pressing the down arrow on the keyboard. Notice the > now points to Constraint. Using the up and down arrows move the > to the game you wish to play and press the SELECT key. This tells the gameshell you want to play that game. The SELECT key 'selects' an item from a menu.

The information on how to use a menu is reflected in the prompt line which states that 'Arrows move the cursor', which is the name for the > (which is exactly what happened when you pressed the DOWN arrow key, the > moved down), 'SELECT to select a game', (which is also what happened!), QUIT which just starts the gameshell all over again and HELP which will provide a short description of the gameshell.

After a game is finished, the system returns to the gameshell and another game can be selected. Pick a game and get started!!!!

## Section 2
## The Terak Games

### Choosing a Database

To expidite learning, the material to be learned has been subdivided into a number of levels. The highest division is that of the databse level. If, for instance, the Gaming System is being used to use geography, there would most likely be a databse for each region or possibly continent, depending on how much information was to be covered. The student must first choose a database.

At the start of most games, a menu appears listing the databases available for this game. If there is only one database on the system, the games will automatically choose it. In the menu listing the databases, move the cursor up or down with the up and down arrow keys until the cursor is in front of the correct database and then press the SELECT key.

An example database menu might contain:

```
Databases:

 ____________________________________________
 | Horses - breeds of horses                  |
 | Presidents - U.S. from 1900 to present     |
 | Authors - 20th century fiction writers     |
 | States - The United States geography       |
 | Uncommon Knowlege - trivia                 |
 |____________________________________________|
```

The prompt line then tells what operations are available with these databases:

```
ARROWS move cursor, press SELECT to select a database,
  QUIT to quit
```

In a selected database, there are subcategories allowing still more specialization of focus in the games. After selecting a database, a new menu appears, entitled : 'Categories', with the word: 'object' as the first element of the menu. ~~If~~ for instance the database 'Uncommon Knowlege' had been selected, the menu might contain:

```
object
  animal
  vegatable
  mineral
```

These are subcategories of 'Uncommon Knowlege'. If the entire set of common knowlege is to be used, just press the SELECT key after moving the cursor in front of object. If on the other hand, animals are to be studied, use the down arrow key to move the cursor in front of animal, and press SELECT.

After something has been selected, the prompt line asks:

> Would you like to choose a more specific item of that category[YES/NO]?

which gives the opportunity to study a subcategory of the selected item, which in this case would be animal. If NO is pressed, the game will be played with all animals. If YES is pressed, a new menu will come up:

```
animal
  nonhuman
  human
```

One of these subcategories can now be chosen with the oportunity of continuing to refine the area of study. If this is your first exploration into the database, a wise action is to explore around a bit to see what the subcategories are. If you get to a place were you've "gone too far", it is always possible to press the QUIT button which will cause the gameshell to reappear and you can then reselect the same game, this time choosing the appropriate category. Continuing down the "Uncommon Knowlege' database, by selecting human, reveals:

```
human
  cro-magnon
  austrolapithacus
  Zaa Zaa Gabor
  etc.
```

And then if Zaa Zaa is selected the choices would be:

```
Zaa Zaa Gabor
  husbands
  homes
  yachts
```

and an interesting game could be played from this point.

There are two constraints on picking subcategories of a database. It is possible to select too specific a category for some games. If this happens, the game will give a nasty message and a less specific category will have to be choosen. In other games, if too broad a category is choosen, the game may be a bit slow. After playing the games a while, you will develop a feeling for the range of proper specification.

## Concentration

You have selected Concentration. The system responds:

Would you like instructions for this game? [YES/NO]

As noted in the introduction, every time a new part of the system is entered a quick explaination is available. By pressing the YES button, a quick description of Concentration can be obtained. By pressing the NO button, we continue.

At this point, a database and category of that database must be chosen. See the above discussion: "Choosing a Database" for instructions. After these selctions, the game initializes (which takes a few seconds) and then the game begins.

As the game comes up, on the left side of the screen apears a menu divided into eight boxes numbered one to eight. Instructions are on the lower right, and the score boxes on the upper right. Behind each box is an item. To see an item, the box must be turned over. Using the up and down arrow keys move the cursor in front of the box to be turned over and press SELECT. The item appears.
The idea of this game is to match items that are directly related. If for instance, we were in a database of U.S. Presidents, and we had a square that said:

George Washington

and another that said:

First president

then since Washington was the first president these match. To indicate this to the game, press the ANSWER key . The challenge of the game is to find and remember where the various items are.

When the game starts all the boxes are hidden . The first box is turned over by pressing the SELECT key. So, using the president database, we start by selecting the first box. Doing this the following appears:

Lady Bird

Notice the bonus score box. It has decreased by five. It decreases by five each time a box is flipped. You move down to the next box using the down arrow key and press SELECT:

Thomas Jefferson

appears. These do not match. IF you press the ANSWER key you will see:

Sorry, that's incorrect. .

Illustrating that there is no relationship between Lady Bird Johnson and Thomas Jefferson. Notice that for this incorrect match a penalty of five points is deducted from the score.

Now try to turn over another square so you can match that one. It doesn't work, because you can only have two boxes turned over at a time. You must turn at least one of the boxes (Lady Bird or Thomas Jefferson) back over. Say you turn Lady Bird over by moving the cursor in front of Lady Bird and pressing SELECT. Notice that the bonus decreases by five, the score by two. It is becuase of these penalties that forgetting where items are located decreases your score. Now move the cursor to positon three and press SELECT. This turns this box over and you see:

Monticello

This of course matches since it was Jefferson's home. So press the ANSWER key. At the box at the bottom it says

That's correct . . . Thomas Jefferson's home is Monticello

Notice the score increases by ten. Boxes can match more than one other box. If a given box only matches one other box , say Monticello only matches Jefferson. After it is matched , a 'clue' appears in the box. If on the otherhand, a box matches multiple boxes, the 'clue' appears only after all the matches have been discovered. In the above example with Jefferson and Monticello, after the ANSWER key is pressed the 'That's Correct' prompt will appear and a clue will appear under Monticello:

* married in office

The Thomas Jefferson box will still show:

Thomas Jefferson

From this, it is possible to conclude that Thomas Jefferson matches another box. At this point there is no score penalty for flipping Jefferson back over. As you continue to play, under box number seven is:

Declaration of Independence

If the Thomas Jefferson box is still turned up and ANSWER is pressed the prompt states:

That's Correct . . . Thomas Jefferson wrote the Declaration of Independence

Under each matched box there is a clue or if you are unlucky it will say:

**********No Clue, Clyde**************

These clues are revealed after all the matches the box participates in are discovered. Taken together, all the clues point to one object in the database. This is called the bonus item. If it is correctly guessed, the bonus score will be added to score to give the final score. At the end of the game, or if you are brave before the end of the game(By pressing the NEXT key) the game will end and you can guess the bonus question. If you accidentally hit the NEXT key during the play of the game, you can return to the game by just pressing the NEXT key again. If you know the bonus item during the game and wish to guess it, press NEXT key and the prompt line will state:

```
ANSWER to guess bonus, HINT for choices, NEXT to return
        or QUIT
```

which states that amoung other things you can press NEXT to get back to the game or QUIT out of the game entirely.

If you make it to the end of the game, discovering all the matches, then you also must answer the bonus question. At the end of the game, the prompt gives the following choices:

```
Type ANSWER to guess the bonus question, HINT for
        choices, or QUIT.
```

Obviously, it is possible in either case to QUIT, but this doesn't help your score much!! There are two ways to guess the bonus item. The first of these is to type the correct answer in. To do this, press the ANSWER key. The system prompts:

```
What is your guess?
```

At which time the correct answer should be typed in. If on the other hand, you find that you have an idea, but would like to choose from a list of possible answers, press the HINT button. A menu called Bonus List appears in the lower right corner of the screen. Use the up and down arrow keys to position the cursor in front of the desired answer and press SELECT. If you were wrong the system responds:

```
That isn't correct. Type NEXT to continue
The correct answer is : Grover Cleveland
```

Since if you remember one of the clues was

```
* married in office
```

and Cleveland was the only president married in office.

If you had in fact typed Cleveland, then the system would say:

```
You answered it correctly!! Very good!! Type
        NEXT to continue
```

After pressing NEXT in either case the system asks:

```
Would you like to play again[YES/NO] ?
```

At which time, you have the opportunity to play this exciting game once again!!

Since scoring is slightly confusing, here is a short summery. Obviously, the object of the game is to get as high a score as possible. The game has two score positions, the SCORE and the BONUS positon. The BONUS position tells how many points are available if you guess the bonus question correctly.

The SCORE position keeps a running track of your success record. You receive 10 points for each correct match, -5 points for each incorrect match and the sneaky part is that you loose 2 points for each unflip (turning back over an exposed box), unless the item was just correctly matched in which case there is no score penalty for unflipping the box. In this way, you are penalized for just randomly turning over boxes.

The BUNUS position decreases by five every time you turn over a box, but will not decrease below 5. The strategy then is to match items and answer bonus question in as few moves as possible.

When you get to the end of the game, or press the NEXT key during the game, and guess the bonus item. The bonus score is then added to your score (assuming you got the bonus item) giving the final score, your score for the game.

## Constraint

You have selected Constraint. The system responds:

Would you like a description of the CONSTRAINT game[YES/NO]?

As noted in the introduction, every time a new part of the system is entered a quick explaination is available. By pressing the YES button, a short description of the constraint game can be obtained. By pressing the NO button we continue.

At this point, a database and category of that database must be chosen in order to play Constraint. See the above discussion: "Choosing a Database" to do this.

The final question before the game starts is:

Would you like to be timed on your answers[YES/NO]

This provides the option of being timed while the game is in progress. This makes the game more exciting and in the timing runs it is possible to get higher scores than without timing.

In this game, the computer provides a list of objects. It then askes questions. Your job is to select from the list of objects, those things which answer the question. If for instance, the database was famous Americans in literature, the menu of ojects might contain:

Hemingway
Frost
cummings
Gardner
Cartland
Falkner
Plath

If you were playing with this database, these items would appear in the left menu entitled, Possible Objects. The first question would appear above this menu. A possible question would be:

What committed suicide?

If you answered YES to timing, a clock will appear in the middle of the screen and will begin to increment. You must now move up and down in the Possible Objects menu using the up and down arrow keys and pressing the SELECT key when the cursor is in front of those persons in the list who had committed suicide.

Let us say you selected Frost, cummings, Gardner and Plath. Notice that they are highlighted when selected. Suddenly you remember Gardner died of a heart attack. Just move the cursor back in front of his name and press the SELECT key again. His name is no longer highlighted and no points have been lost, but time is fleeting by.

To stop the clock and evaluate the selections press the ANSWER key.

The system will now evaluate the choices. The Possible Objects menu will now contain objects marked in three ways. First will be the hightlighted objects. These are the people who committed suicide and were selected as having done so. The second type are those that were selected as having committed suicide and didn't (they are angry at you!). This second type of person is marked with a =>. The last type of person are those who committed suicide, but were missed. These people are marked with a ->. All other items are removed from the menu.

In this case, Plath would be hightlighted, since she was selected and blew his brains out. Frost and cummings would appear:

```
=>cummings
=>Frost
```

since both of them died of ripe old age, but had been selected. Finally, Hemmingway would appear as:

```
->Hemmingway
```

Since he was not selected, but blew his brains out. All the other entries (including Gardner which was selected and then unselected) are gone.

The prompt line at the bottom of the screen now states:

```
ARROWS move cursor, Press NEXT to continue
```

When NEXT is pressed, all persons who did not commit suicide are eliminated from the menu, leaving only Plath and Hemmingway. The description: committed suicide, is placed in the Description menu on the right of the screen along with your score for this question and the time it took to answer, if timing was turned on. At the bottom of the screen, the cumulative points for this set of objects is displayed. The prompt now states:

```
Press NEXT for next question
```

The cycle of questions appearing and selecting objects repeats. The cycle terminates when the constraints uniquely identify the
In the case of the authors, the next question may be:

```
What's profession was novelist?
```

This is of course Hemmingway, as Plath wrote poems. Selecting Hemmingway and going through the answer business, eliminates Plath, leaving only Hemingway. The system now asks:

```
Would you like to play again[YES/NO]?
```

Which gives the opportunity to get another set of objects and/or questions.

If for some reason the objects in the menu are identical as to the information placed in the database the system will respond:

Can't distinguish remaining items,Press NEXT to continue

After pressing next the prompt:

Would you like to play again[YES/NO]?

will appear.

The scoring for this game depends on whether timing has been selected. If timing has not been selected, 1 point is received for each item correctly selected, 1 point is subtracted for either incorrectly selecting an item or not selecting an item and it is correct. If timing has been selected, the penaltys are the same but the method of awarding points is more complex.

The speed at selecting answers determines the number of points available for each correct answer. If one right answer is selected every two seconds each right answer receives four points. If a correct answer is selected on average every four seconds each right answer receives three points, if a correct answer is selected on average once each six seconds each receives two points, otherwise it is the same as without timing, where a correct answer receives one point. Note that with timing if the time allotted runs out, the system will start evaluating the answers.

## Flash

You have selected Flash. The system responds:

    Would you like a description of the Flash game[YES/NO]?

As noted in the introduction, every time a new part of the system is entered it is possible to get a quick explaination. By pressing the YES button, a quick description of the Flash game can be obtained. By pressing the NO button, we continue.

At this point, a database and category of that database must be choosen. See the above discussion: "Choosing a Database" for information on this task.

In the traditional flash card game, a card is pulled from a deck of cards. On one side of the card is a question, on the other the answer. The individual pulls card, reads the question and answers the question. The card is then turned over to see if the correct answer was given. The computer game is similar. The computer asks a question, an answer is given and the computer checks to see if the answer was correct.

When the game comes up, a menu entitled correct is displayed on the screen. It is here that correct answers are stored. Across the top of the screen is scoring information totals. At the beginning of the game the situation is:

```
Totals: 0 Correct; 0 Wrong; 0 Hints; 0 Answers Given;
```

A question appears on the left under "Totals". If the database in use was States of the United States, the question may be:

```
            New York's capital is what?
                 1 answer left
```

The prompt gives the following options:

```
            ANSWER to enter an answer, Arrows move cursor, QUIT,
              HELP,HINT-gives possibilities, TELLME gives an answer,
              NEXT-next question
```

The "1 answer left" under the question, tells that there is only one correct answer for this question. There are four ways to deal with this question. The first is to type in the answer. To do this, press the ANSWER key. A cursor appears on the left of the screen. If the correct answer: Albany is typed in followed by RETURN being pressed, Where the "1 answer left" was appears:

```
            Correct
```

Then almost immediately appears

```
NEXT for next question
```

in the same place. This indicates that the correct response was given. Notice also that in the Totals at the top of the screen:

    1 Correct

appears. Pressing NEXT gives the next question:

    What borders California?
    3 answers left

A second method of giving an answer is by pressing the HINT key. A menu of possible answers appears on the screen. In this menu are possible answers to this question. If the only possible answers are correct, or this is a number question the system will not give a Possible Answers menu. In this case, a partial contents of the menu shows:

    Ohio
    New York
    Oregon
    Colorado
    Nevada
    New Mexico
    Hawaii
    Alabama
    Arizona
    Alaska

Using the up and down arrows it is possible to move the cursor in front of the correct choices. If in this case, the cursor is moved in front of Hawaii and SELECT is pressed, Hawaii becomes highlighted. The system then checks if the highlighted item answers the quesion. Under the question the phrase:

    incorrect

appears, since Hawaii doesn't border California. If Nevada is selected the phrase:

    correct

appears and Nevada moves from the possible answers to the correct menu, since Nevada borders California.

Another way of getting an answer is by pressing the TELLME key. This key gives one answer. Both TELLMEs and HINTs are noted in the scores at the top of the screen.

Finally, if a question comes up that is uninteresting, press the NEXT key and another question will come up.

When all the questions in a given area have been answered correctly the system comes up with:

    There are no more questions in this category.
    Press NEXT for new category, QUIT to quit.

## Twenty Questions

You have selected Twenty Questions. The system responds:

    Would you like a description of the Twenty Questions
        game[YES/NO]?

As noted in the introduction, every time a new part of the system is entered a quick explaination is available. By pressing the YES button, a short description of the TWENTY QUESTIONS game can be obtained. By pressing the NO button, we continue.

At this point, a database and category of that database must be chosen in order to play this game. See the above discussion: "Chosing a Database" to do this.

In this game, items from the category of the database that was chosen are presented. The computer has selected one of the items, (the computer is 'thinking' of one of the items). The idea is to eliminate items from this list until only the item the computer is "thinking about" remains. This elimination is accomplished by choosing attributes or characteristics that are shared by some subset of the items. The computer will take the characteristic, see if the item it is thinking of has it. If it does, then the computer eliminates from the list those items that do not have the characteristic.

If the hidden item doesn't have the chosen characteristic, then items that do have the characteristic are eliminated from the list. So, by carefully choosing characteristics the list is narrowed down to the item the computer is thinking of.

As an example consider a database of U.S. States. If the category 'important states' was chosen the following states would likely be in it:

California
Florida
Iowa
Ohio
New York
Maine
Massachusets
Minnesota
Nevada
Virginia
Washington

and possibly others... The computer is thinking of one of these states and the task is to narrow the list down. In the game, possible attributes are given but for the moment consider how this list could be narrowed down. As a first cut, we can divide those states that are on the atlantic ocean versus those that are not. If the computer had an atlantic state on its mind, then it would eliminate California, Iowa, Ohio, Minnesota, Nevada, and Washington. This leaves:

Florida
Massachusetts
New York
Maine
Virginia

As a next cut, note that some of the states are of the original 13 colonies while others are not. If we use this cut, and the computer was NOT thinking of states that were of the original 13, then Massachusetts, New York, and Virginia would be eliminated. This leaves:

Florida
Maine

Finally notice that Florida's major crop is citrus fruit, while it is not of Maine. If this is the cut and the computer was thinking of Florida then the task is over.

It is clear that the type of characteristic that is optimal is one that divides the list into two approximately equal parts, those with and those without the characteristic. In the above example, the first characteristic was 'atlantic' with 6 of the items eliminated and 5 of the items not. So when it was discovered that the computers item wasn't on the atlantic about half of the items were eliminated. If on the other hand, the computers item had not been on the atlantic, still about half of the items would have been eliminated.

Similarly at the second cut, the characteristic was 'original 13 colonies'. There were 3 yeses and 2 no's. No matter whether the item the computer selected was on "of the 13 colonies" or not about half of the list would have been eliminated. Similarly down the line till only Florida was left.

The manner in which the game is played closely resembles this mental experiment. Again using the States database, consider how the computer game would be played.

When you have selected the database you want to work with, the computer will show you a screen with three boxes; one labelled "Remaining Items" on the left side of the screen, one labelled "True of Hidden Items" on the upper right side, and a box labelled "False of Hidden Items" on the lower right side. These two right hand boxes will be explained in a moment.

First, let's examine the box labelled "Remaining Items". One of the items in this list is the item the computer is "thinking of". Your job is to find that item by eliminating all the other possible items. To eliminate the false items you ask questions of the computer about the characteristics of the item it is "thinking about". Therefore, the first task you want to do is study the list of "Remaining Items" to determine which characteristic you want to ask about. In the case of the "Important States" database, one feature you might want to ask about is whether the state the computer is "thinking about" borders on the Atlantic. Once you have decided on a characteristic you want to know about, press ANSWER.

After pressing ANSWER, the screen will clear and you will see a box labelled "Possible Questions" with a list of questions with blanks either at the beginning or end. Here are some of the forms of questions you will see:

__________ borders it

It borders __________

The first question is asking "Does __________ border the state the computer is "thinking about?" The second question is asking " Does this state the computer is "thinking about" border on __________?"

__________ is the major crop of it.

It has a major crop of __________

The first question is asking Is __________ the major crop of the state the computer is "thinking about?" The second question is asking " Does this state the computer is "thinking about" have a major crop of __________?"

Another example might be:

It has __________

This question is asking "does the state the computer is "thinking about" have a __________?"

Move the cursor to the question form you want to ask and press SELECT. The question you have selected will appear at the top of the screen with the cursor in front of the blank. You must fill in the blank with the characteristic you are asking the computer about. For instance, in our example we wanted to determine whether the state the computer is "thinking about" borders an ocean, we would select the question form

It borders a(n) __________

and then type in the word "ocean". Then press RETURN.

After pressing RETURN, the screen will clear and you will again see the screen with the three boxes labelled "Remaining Items", "True of Hidden Items", and "False of Hidden Items".

At the bottom of the screen you will see the word "checking" flash on the screen. This means the computer is determining whether your question has anything to do with the item it is "thinking about". In our example, the computer is determining whether the state it is "thinking about" borders an ocean. The next operation the computer performs depends on one of the two following conditions:

1) THE QUESTION ASKED IS RELATED TO THE HIDDEN ITEM THE COMPUTER IS "THINKING ABOUT". (In our example, the state the computer is "thinking about" borders an ocean) In this case, the question is rewritten in the "True of Hidden Items" box and is highlighted. Then the computer scans the items in the "Remaining Items" box and highlights all those that share the same characteristic asked for in your question. For instance, in our example, all the states that border the ocean would be highlighted. Finally, all the remaining UNHIGHLIGHTED items are marked by the computer for elimination by placing a --> in front of them.

2) THE QUESTION ASKED IS N O T RELATED TO THE HIDDEN ITEM THE COMPUTER IS "THINKING ABOUT". (In our example, the state the computer is "thinking about" does NOT border an ocean) In this case, the question is rewritten in the "False of Hidden Item" box and is highlighted. Then the computer scans the items in the "Remaining Items" box and highlights all those that share the same characteristic asked for in your question. For insatnce, in our exampe, all the states that border an ocean will be highlighted. Finally, all the HIGHLIGHTED items will be labelled for elimination by placing a --> in front of them.

Once the computer has highlighted and marked the appropriate items in the "Remaining Items" box, press the NEXT key and only the items that do NOT have a "-->" in front of them will remain. One of these remaining items is the item the computer is "thinking about". To narrow the remaining items even further, you continue to ask questions until you have determined the "hidden item".

## Jeopardy

You have selected Jeopardy. The system responds:

Would you like a description of the Jeopardy game[YES/NO]?

As noted in the introduction, every time a new part of the system is entered it is possible to get a quick explaination. By pressing the YES button, a quick description of the Flash game can be obtained. By pressing the NO button, we continue.

At this point, a database must be chossen. See the above discussion: "Choosing a Database" for information on this task. However, to explain this game we want you to use the AUTOBASE. So put your cursor nest to the item AUTOBASE and press SELECT.

The object of this game is to generate the right question from certain facts given about an unknown item. The first box that appears after you have selected your database includes all the categories that the computer knows about the AUTOBASE. You must select five of these categories from the list. For our example, SELECT the engine part, body part, external-frill, safety part , and drive-train part categories. There is a counter on the right side of the screen to keep track of the number of categories you have selected. Remember, YOU CAN SELECT ONLY FIVE CATAGORIES. After the category selection has been made, press NEXT.

After the answers are loaded, you see a screen with five columns of five boxes apiece, each numbered from 10 through 50. Above each column there is the category name signifying the subject matter to be tested in each column.

The game is setup so that the player must answer the box labelled "10" before your can answer the box labelled "20", and the player must answer the box labelled "20" before he answers the box labelled "30", and etc. through "50". The player can move from one category to another at any time.

Now move the cursor to the external-frill category and press the SELECT key.

The screen clears and you are presented, in a box in the middle of the screen, with one or more true statements about a certain external-frill. If this box as the answers:

> It functionally connects to radio,
> and it is a type of external-frill

then we can continue with our example. If it does not, then press first the RETURN key TWICE and you will return to the display with all the boxes. SELECT the next box in the category "external-frill". Continue selecting boxes in this category until you get the right answers in the "answer box".

Now we have the right box with the answers:

It functionally connects to radio,
and it is a type of external-frill

Below the box there is a question in which you must type in the name of the item which best fits the description given in the box above. The question in this example is:

What is (a) ? (type in answer).

Now you must complete the question above with the name of an item that best fits the answers in the above box. What item is both an external-frill and functionally connects to (which means it connects by a wire or cable, etc.) the radio? In this case the term needed to complete the question is "antenna". So type in "antenna" and press RETURN. You should see on the screen below your answer this message:

Correct!
Press any key to continue

If you happen to complete the question with an incorrect term, the message at the bottom of the screen will read:

No, I'm sorry, the question was: What is (a) antenna?

Press any key to continue.

After pressing any key, you will return to the display with all the remaining boxes, so that you may select again.

Your score is tabulated after every turn in the upper right corner of the screen.

After you have selected all the boxes, you get one final jeopardy category where you can bet up to your total current points on your ability to complete the question with the correct term. First you type in your bet and hit RETURN. Now complete the final jeopardy question. Your bet is either subtracted, if you were wrong, or added, if you were right, to your current score to give you your final jeopardy point total.

## The Component Identification Game

The component identification game, alse called the parts quiz, is designed to drill you on where an component is on a object it is part of. To do this, you will be presented with a picture of an object, and a particular portion of that object will be indicated, and you will be asked to identify the component that is at that position.

When the game starts up, if there is more than one database with parts information, you will be presented with a list of available databases, otherwise the game will load the one available database and proceed (a parts database has a different structure than the databases the other games use, so there may not be a parts database for every ordinary database on your gaming system). If a menu appears, asking for you to select a database, use the up arrow and down arrow keys to move the cursor in the menu to the database you wish to be tested on, and press the SELECT key to select that database, or press the QUIT key to return to the game shell. The program will clear the screen and write 'Working...' followed by several dots as it loads a database.

Once the program has loaded a database, you will see the words 'Choosing a picture' appear on the screen, followed by several dots as the computer works. Once a picture has been chosen, it will be displayed on the screen, along with the words 'Loading the menu', appeaingr on the next line, also followed by a row of dots as the computer works. Once the computer has finished loading the menu for that picture, the screen will be cleared, and a menu will be displayed in the upper left of the screen containing the components found on the object. The title of this menu is the name of the object selected. In the upper right of the screen is the score display, which looks something like a menu, but has three fields in it, labeled 'Right', 'Wrong', and 'Told'.

These fields give the totals for the questions you got right, wrong, or had the computer tell you. Next, the computer will write 'Choosing a question' near the top of the screen, followed by a row of dots as the computer works. This line will be replaced by the message 'Identify the component next to the asterisk', along with a flashing asterisk ('*') somewhere on the picture. The computer will then write the message 'SELECT an object, NEXT picture, TELL ME an answer, QUIT', describing your command options at this point in the game. In the upper center of the screen, ~~the~~ computer will tell you how many correct responses there are left for that position. If it says '1 answers left', only one item in the menu matches that position. If it says something like '3 answers left', that means that there are three items in the menu that match that position, and all three will have to be identified before proceeding to the next position.

To identify the item at that position, use the up and down arrow keys to move the cursor along the component menu to the component you think belongs at that position and press the SELECT key to tell the computer your choice. The computer will tell you whether you are right or wrong, and update your scores and the correct answer count. If there are still more answers for this position, or if you selected a component not at that position, you will then be able to select another component.

The components you selected before remain complemented to tell you that you have already used those answers. If you try to select one of them, the computer will tell you that you have already selected that answer, and will refuse to score it. Once you have correctly identified all components at a position, the computer will display the message 'Press NEXT to continue, QUIT.'. Pressing the NEXT key will cause the computer to go on to the next question, and pressing the QUIT key will cause you to leave the game.

If you have no idea what component belongs at that position on the picture, you can press the TELL ME key, and the computer will tell you what the component at that position is, complement that item in the component menu, and update the 'Told' count and the correct answer count. The computer will then display the message 'Press NEXT to continue, QUIT.'. Pressing the NEXT key will go on to the next question, or, if there are still more answers for this position, once more ask you to identify the component at that position. Pressing the QUIT key will cause you to leave the game.

If you decide that you wish to be tested on another picture, when the computer asks you to select the object at a point on the picture you may press the NEXT key, and the computer will clear the screen and choose a new picture, as described above. If you decide you want to exit the game, you may press the QUIT key, and the computer will display your final scores and ask you to press the SPACE bar to exit the game.

The computer will not choose a new picture until you have correctly identified all the components of a picture the first time (for example, if you made a mistake on a component, then answered it correctly, the program will ask about the same position later on, and will do it again and again until you answer it correctly the first time). Once this is done, the computer will automatically clear the screen and choose a new picture. The computer checks your input, and will make sure that you know a picture before letting you forget about it.

If you make the computer choose a new picture by pressing NEXT, or if you made a mistake on a picture before answering it correctly, it remembers that picture, and will choose it again (and again and again) later in the program, until you get everything right the first time. When the program runs out of pictures (because you've answered everything right on each one), or you have quit the game from some point before finishing, it will clear the screen and display your final scores, and display the message 'Press SPACE to continue.'. Pressing the SPACE bar will cause the program to exit to the game shell, where you may choose another game to play.

## Show

You have selected Show.

Show is not really a game. It is more like a database examiner, where you see can how the items of the data base are related to each other. To learning how to use SHOW we will pick a database on the system and try to obtain some information about certain items in it.

Lets say in this example we want to know what relation "transmission" has to the rest of the parts of a car. Thus, in this case we are going to use the "AUTOBASE". So move the cursor down to the menu item "Autobase" and press SELECT.

The next line you see in SHOW is as follows:

Do you wish to use the menus? [YES/NO]

For this example we want to use the menus, so press YES.

In the display you see now, the top of the box is labelled "objects" and the items listed inside the box are the sub-categories of "OBJECT". In this database there is only one term, "part", but in the "Soviet Navy" and the "Star Trek" databases there are three catagories under object: weapon, platform, and equiupment.

The lines at the bottom of the screen tell you the commands you can use and what database your are currently using.

Since there is only one choice for us in this first box, just press SELECT. The next display shows you what sub-categories are under the category "part". Let's read this diagram starting from the upper left-hand corner. The first circle says "SUSPENSION-PART" and has a arrow draw from its circle to the circle termed "PART" in the middle of the screen. This line is labelled "ISA" and has a arrowhead pointing towards the "PART" circle. You can interpret the meaning of this diagram of the two circles and the arrow as follows: "SUSPENSION-PART" is a "PART". This means that suspensions parts belong to the category "PART". We can also see that all the other circles, except one, surrounding the "PART" circle all have arrows pointing towards the PART circle. Thus, all those categories are also sub- categories of the category "PART".

Now look at the circle that is on the bottom of the screen, second from the right, labelled the "OBJECT" circle. Notice which way the arrow is pointing. It points from the PART circle TO the OBJECT circle. This means that "PART" is a sub-category of the category "OBJECT". A good way to think interpreting these arrows is the following:

THE ARROW POINTS TO THE TERM THAT IS BEING DESCRIBED.
THE OTHER TERM IS DESCRIBING SOME ASPECT OF THAT TERM.

Getting back to our transmission example, down in the lower left corner of the screen the prompt line tells us to "press NEXT to continue." Do this when you understand what the present display is showing.

When you have pressed NEXT, you see a new box on the right side of the screen. The box is labelled "Linked to Part". This means that the nine sub-categories listed in the box are linked to the category PART. These are the same sub-categories that you just saw in the previous screen, but appear in a menu rather than a graphic format.

We are looking for the category that has the item "transmission" in it. Why don't we choose "DRIVE-TRAIN PARTS" as our category. To do this move the cursor next to the term "DRIVE-TRAIN PARTS" in the box above and press SELECT.

In the next display we see all the items that are considered DRIVE-TRAIN PARTS in graphic display. Notice all the arrows are pointed towards the middle circle labelled "DRIVE-TRAIN PART" except the circle on upper part of the scrren, two in from the right. This circle is the "PART" circle in which DRIVE-TRAIN PART is a sub-category. We also discover that the big cirlce in the upper right corner of the screen is labelled "TRANSMISSION". This is the part we are looking for so we know we have the right category.

After we are finished looking at this display, press NEXT as per instructins at the bottom left corner of the screen. Now we see similar dispaly as before, except the right box is labelled " Linked to Drive-Train Part rather than "Linked to Part". Also, the list of items inside this box is of just drive-train parts and not the names of different categories as was the previous one. Since we are interested in the itme "TRANSMISSION", move the cursor to the word transmission and press SELECT.

This current display shows all the relationships the item TRANSMISSIN has to the rest of the database. Lets look at each circle one at a time.

Upper left corner: this circle is labelled "DRIVESHAFT" and has arrow pointing from the center circle labelled "TRANSMISSION" towards it. Now notice how the line is labelled. It is not the word "ISA" but the word "CONNECTS-TO". This symbol means that the transmission is connected to the driveshaft.

| | |
|---|---|
| Upper center: | This circle is labelled "FRAME" and the arrow is pointing towards it from the center circle and is labbeled "MOUNTS- ON". This means that the transmission mounts on the frame of the car. |
| Upper right: | This circle is labelled "KEEP ENGINE IN PROPER RPM RANGE". The arrow is labelled "USE" and is pointing away from the center circle. This diagram means that the transmission is used to keep the engine in the proper RPM range. |
| Lower right: | We have seen this before. It means that transmission is the sub-category of the category DRIVE-TRAIN PART. |
| Lower middle: | Unfortunately, this is very hard to read, but the circle is labelled "CLUTCH". The arrow arrow is pointing towards the center for the first time in this current display, and the line is labelled "FUNCTIONALLY CONNECTS-TO". This means that the clutch is not physically touching the transmission when it connmects to it, but it connects to it through a wire, or tube, or in this case by a cable. |
| Lower left: | This is also very unclear, but the circle is labelled "SHIFT LEVER". The arrow is pointing towards the center circle and is labelled "FUNCTIONALLY CONNECTS-TO". This means that it is connected to the transmission, but not physically touching the transmission. |

Now let's return to the top menu again. To do this press NEXT, then press QUIT. Now SELECT SHOW again and also SELECT the AUTOBASE. On this screen you are again asked:

Do you wish to use the menus? [YES/NO]

In the previous example we did use the menus, but in this case we will not. So press the NO key.

Now you see on the screen this question:

What item do you wish to show (QUIT to quit) ?

Let's say we are still want to know about the 'transmission' and how it is related to the other parts of a car. In the previous example we had to see the graphic display of all the various categories above "transmission" before we found the term we wanted. If we do not use the menu format, then we just type in the part we want to know about and the system will take you to it. To see what I mean, type in the term TRANSMISSION and press RETURN.

Here we are viewing the "transmission" graphic display without having to go through all the intermediate steps. When you press NEXT, it returns you to the display that says:

What item do you wish to show (QUIT to quit) ?

Now you have the option to select another item to display or you can quit SHOW.

If it takes fewer steps and time to not use the menus, why should anyone use them? If you are unfimiliar with the database you are using, the menu option shows the organization of the database and where the item you want to know about is placed in that database. If you already know the basic organization of the database and only want to know about specific items, then don't use the menus.

While playing these games you will use the keyboard to interact with the computer. Always look at the prompt line, located at the top or bottom of your screen, to know which keys are available. Right now, for example, you can see that NEXT (located on the right side of you keyboard) is available when you are finished reading this. Press NEXT now.

{PAGE}

We will now give you a very brief description of the keys. As before, press NEXT after you are finished reading each item.

{PAGE}

Arrow Keys:

These move you position locator (cursor) around the screen. Use the arrows to move up and down a list of possible answers (in other words, in a multiple choice format) to position your cursor in front of the correct one. Arrow keys will also scroll information when there is too much information to be displayed in one box. When you see the up-arrow symbols (^^^^) press the up-arrow key on your keyboard (or the down-arrow key when you see down-arrow symbols) -- this will bring up the rest of the display for you .

{PAGE}

SELECT:

Press this key to indicate that you are choosing the answer you cursor is next to.

{PAGE}

ANSWER and RETURN

You will not always answer a question by SELECTing from a list of possibilities. Sometimes you will have fill-in-the-blank type of questions and it will be necessary to type in you answer. Before you do, press the ANSWER key to alert the computer to recieve your input. After typing in your answer, press RETURN to signal the computer that you are finished. Once again the order of events a fill-in-the-blank question are:

1. Press ANSWER
2. Type in your answer
3. Press RETURN

{PAGE}

SPACE, BACKSPACE, number, and letter keys:

When it is necessary to type in an answer, use the keys as you would on a standard typewriter keyboard -- numbers and letters in your answer, SPACE between words, and BACKSPACE to correct an entry.

{PAGE}

TELL ME:

This is a "give up" function, it will give you the answer. When there are multiple answers it will give you one answer for each key press.

{PAGE}

HINT:

This function, where available, will give you information to help you answer a question.

{PAGE}

QUIT:

This key will always allow you to terminate what you are doing at any given time.

{PAGE}

HELP:

Pressing this key will give you infomation to facilitate your interaction with this gaming system. You will get different information at different places in the system.

Pressing HELP at the first page in the system (the one that has the list of all the games) will give you several help options to choose from including this description of the key functions.

Pressing help while in a game will give you a description of that game only.

Figure 1

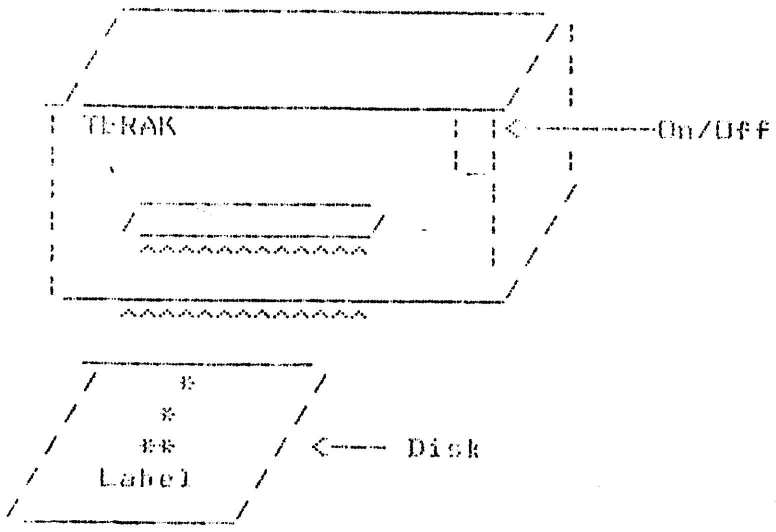

Figure 2

```
Welcome to the NPRDC Tactical Gaming System[5.2]
Hello, Sir


Select one of the following:
 _____________________________________________________
|                                                     |
>  Concentration (with apologies to Hugh Downs )      |
|  Constraint - the game of narrowing choices         |
|  Flash - an interesting way to spend some time      |
|  Parts Quiz - the visual stimuli response game      |
|  Picture Quiz - the game of pattern recognition     |
|  Show - a graphical means of information retrieval  |
|  Twenty Questions - a game of knowlege              |
|  Jeopardy - The Answer Game                         |
|_____________________________________________________|

ARROWS move the cursor,SELECT to select a game,QUIT when done, or HELP
```

## NPRDC Gaming System 5.2
## Constructing Databases

## SECTION 1
## INTRODUCTION

I think I shall never see
A database as lovely as a tree.
Anonomous

NPRDC has a continuing interest in instructional systems that do not require the questions to be "hard wired" in. In many respects, the NPRDC 5.2 gaming system accomplishes this objective. The games of the system look to a database to provide them with information to ask questions and determine if the responses are correct. Hence, constructing a reasonable database is important if the games are to function correctly.

The games expect a semantic network database. A network consists of nodes which represent concepts and links betweens the nodes. Semantic networks are naturally suited to computer based instruction because the computer can easily browse along the links to construct exercises and instructions on meaningfully related concepts.

This document provides information in constructing a text form of suitable semantic network that can be converted with software provided into a database that the games can use. Section 1A provides information for making databases that do not stray too much from the example databases provided. Four example databases have been provided as templates for new databases. If more bravery is available, a more technical section is provided to allow more freedom in constructing databases. Possibly a section should have been provided explaining the linguistic and knowlege structure constraints on this type of enterprise, but that is for a later release.

Included with this document are four database texts, a startrek database containing facinating facts on startrek hardware, a database on the Soviet Navy taken from "Jane's Fighting Ships", a South American database on countries of South America and an Auto database with information on car components.

Finally, making databases is an art. We have made a few and so now know that you put clay in the kiln and oilpaint on canvas not the other way round. Answering questions is always better than working, so if you get in a bind, have suggestions, get an interesting system crash give us a call or drop us a line. It's lonely here at database central.

Note: Gaming System 5.2 is not a production quality system. We ship the system under that understanding. We cannot and will not assume responsiblity for the accuracy of these documents or the correctness of our system. We try as best we can to keep the noise and errors less than signal strength.

## SECTION IA
## QUICKIE COURSE

I made a database
therefore I am.
Wishes to remain
Anonomous

Being in an incredible hurry to make your first database, you come to the quickie section. Well it ain't so easy. In this section, we will consider the construction of a standard database. To get a feeling for constructing a database, the steps used by the individual who constructed the auto database will be considered.

The text for the database will be constructed using the UCSD Editor. For instruction on its use see Bowles' text. THe text file for the autobase is presented in ApendixE!]. The person constructing a database will create a file similar to the appendix in structure. The only difference is that no line numbers will be typed in. They are present only so that specific lines can be referred to in this document.

a). Objects

The auto database was constructed with the intent of familarizing the student with the parts of a car. The constitients of the database that the games will single out to ask questions about are called objects. All the important car parts are objects in the auto database. A few uninteresting parts were make attributes. Attributes are just not important enough to ask questions about. The radiator and the gas tank are important and so are objects. Their caps are not. Other attributes are colors such as red or green, which in an auto databsse would be attributes, but in a database covering primary colors may be objects.

For the moment ignore lines 1-48 of the database text. Line 49 is:

```
part isa object
```

The word isa is called a relation. Relations are the links between constituents of the database. Line 48 can be thought of as defining part to be an object:

```
part is an object
```

The word object is a built-in of database construction. If a database is about geography, statement 48 would possibly be replaced by:

```
state isa object
city isa object
river isa object
     ect.
```

The auto database is unusually simple in that there is only one type of object, the part. In teaching auto parts, the car is broken down into subsystems, such as steering, engine, etc. In constructing the database, this ordering was considered useful and so was retained.

The parts of the car were divided into mutually exclusive sets: [steering part (51), engine-part(76), body-part(178), interior-part(204), safety-part(248) drive-train-part(291), and suspension-part(321)).

In line 51:

```
          steering-part isa part
```

steering part is "hooked" onto part. A nice property of isa is that if

```
          X isa object
```

and

```
          Y isa X
```

then the system knows that

```
          Y isa object.
```

In other words, by saying steering-part isa part, the system makes a new object called steering-part. Similarly, in line 52:

```
          steering-gear isa steering-part
```

the system will trace back and now know about the new object steering-gear. It is in this way that the sytem learns about new objects.

The fact that statements 49-52 appear in the order they do is important (except for line 50- blank spaces can be added to the file to improve readability). If say line 52 came first:

```
          steering-gear isa steering-part
```

the system would be very confused as it didn't know what a steering-part was. In the beginning, the system only knows object. One must step by step build up from object all the "objects" that the system will use. Later when the program 'learn' is run on this text file if something is not previoulsy hooked to object, an erro r will occur.

This building up of objects from the built-in object must be applied in building a database. If the database is on meals, a possible structure might be:

```
     meals isa object
          breakfast isa meal
             cereal isa breakfast
                 kids-stuff-cereal isa cereal
                      Fruit-Loops isa kids-stuff-cereal
                      Frosted-Flakes isa kids-stuff-cereal
                 hypervitamin-cereal isa cereal
                      Total isa hypervitamin-cereal
             eggs-bacon isa breakfast
          lunch isa meal
              etc.
```

Note that objects of more than one word must either be in quotes

```
"This is silly" isa object
```

or hypenated together

```
This-is-silly isa object
```

In the auto database, each of the auto parts is defined using isa. For steering part lines 52-58 define all the steering parts. In lines 60-74, the characteristics of steering parts are listed. To understand these, we must look more deeply at relations.

b) Relations

The only relation discussed so far has been isa. Lines 1-26 set up the basic relations and types of relations. The following section discusses the meanings of these lines in detail. For now, just type these lines in exactly as they appear ( without the line numbers) at the beginning of the text file.

Relations provide links between objects. The relation isa helps to define what are objects. A second relation, same-as helps to define synonyms. Consider:

```
Department-of-Defense isa Executive-Department
DOD same-as Department-of-Defense
DOD in-charge-of defense
```

The same-as in the second line above allows for multiple names of items. If one is playing the Flash game, the question:

```
What Executive-Department in-charge-of defense?
```

might arise. The student could type in DOD or Department-of-Defense and either would be accepted. The in-charge-of in the third line above, brings us to the question of how to define new relations. There are three types of user defineable relations: obj-quantity, attribute and inter-obj- relation.

In this database, an attempt has been made to capture the interrelatiedness of the auto parts. We wanted the student to know what the parts are mounted on (mounted-on), what they are constructed with (constructed-with). Also the student should know mechanical, functional connections (connects-to) and nonmechanical functional connections (functionally-connected-to). The use of the part (use) is also an important relation that the student can should reasonably be expected to be tested about. Finally we used the relation (has) to include notable, but not important parts.

Since the relations:

```
mounts-on
functionally-connects-to
connects-to
```

are relations between parts, they are defined to be inter-obj-relations, which is the type of relations used between objects ( in this case parts). Line 28

    connects-to isa inter-obj-relation

defines connects-to to be a relation of this type. Similarly, in lines 32 and 45 functionally-connects-to and mounts-on are defined as inter-obj-relations.

After these definitions, one can then state relations like line 134:

    thermostat mounts-on cylinder-head

or 142:

    water-pump functionally-connected-to thermostat

The use of a part, what it has and what it is constructed with are attribute type relations. In line 63:

    bellows constructed-with no-metal

it was felt that no-metal was not important enough to ask questions about, so constructed-with is an attribute type relation.

So, in line 36:

    constructed-with isa attribute

defines constructed-with as a relation that doesn't require things on the right (such as no-metal) to be objects. Similarly in line 144:

    piston has rings

rings wasn't important enough to include as a part, so in line 43:

    has isa attribute

has gets its definition.

The last type of relation is the obj-quantity type. This relation is not used in the auto database, but could be. If it is necessary to note the number of an item such as the number of wives of a Brittish king:

    HenryVII married 7 women

would be an appropriate statement. Another example is

    "United States" has 50 states

If the quantity is between 0 and 16383 ( the quantity being 7 and 50 in the two examples above) then nothing special is necessary. If the quantity is out of that range, then it is necessary to make a special type of relation. A case where this is necessary is:

    Nitrogen freezes -209 Fahrenheit

In this case it would be necessary to define:

    freezes isa obj-quantity

which allows positive and negative numbers up to 37 digits long, with six of the digits significant.

There are two assignable characteristics of relations, spelling and importance. Importance gives the games a measure of how important a relation is . Relations of importance 0:

    Not-Much IMPORTANCE 0

are never asked.

Initially attribute, inter-obj-relations and obj-quantity are importance 2. We felt that the relation (use) was not that important, so in line 41:

    use IMPORTANCE 1

Importance can range from 0-2.

The other characteristic is spelling. The games use the spelling relation to construct reasonable English statements from the relations. As an example consider line 40:

    use spelling "%s's use is to %o"

Consider the relation in line 70:

    steering-wheel use "turn the car"

In the Flash game, the question about this relation would be:

    What steering-part's use is to turn the car

Similarly, in the Concentration game, if something is answered correctly the game responds:

    Correct! Steering-wheel's-use is to turn the car

What the games have done is to replace the %s in the spelling template (the "%s's use is to %o") with the subject of the relation:

    steering-wheel use "turn the car"

which is steering-wheel. Similarly the %o is replaced with:

    turn the car

To get a better idea of what this does, consider if use had no spelling. The Flash question would now be:

    What steering-part use turn the car

and the Concentration response would be:

    Correct! Steering-wheel use turn the car

What spelling allows is varied spellings of the relations. The default spelling is:

    "%s %r %o"

which just puts out the line as it appears in the text file. For instance since isa has spelling:

    "%s is a type of %o"

the games know to print out line 52:

    steering-gear isa steering-part

as

    steering-geer is a type of steering-part

which is nicer to read and play the games with. Most of the relations in the auto database have spellings.

Ok so with this knowlege, lets go out and construct a database!!!

## SECTION 2
## TECHNICAL STUFF

| Question: | Answer: |
|---|---|
| What similarity exists between this gaming system and a golfer? | They both play on the links! |

This section requires knowlege from section 1a, so a quick reading of it won't hurt. Here we consider the following points that are not necessary to make a database, but will allow you to ask different types of questions, control which type of questions are asked by some games, and better understand the structure of the gaming system in order to, one hopes, write databases suited for the games.

As stated above, to get a true feel for the games, one must play the games. (examples will be taken from the startrek database, so go out there and play a few rounds) The first aspect of the database that one notices (after you have choosen the database on any game) is that there are levels in the database. That is, when you are asked to select a category, you have the first entry (say "object") and below it you have various types of objects (say "platforms" or "weapons").

This is not accidental! This, the hierarchial nature of the database is controlled by the isa tree. The tree has "object" at the top. For a given thing to be visible, it must be linked to object. This is done using the relation isa, hence the name isa tree. In the space database, there are three objects linked by isa to object: platform, equipment, and weapon. (again note that object is used in many ways, possibly clear from context-object is the thing that is at the top of the isa tree and also things linked to object are called objects-life is tough). So looking through the database one finds:

```
platform isa object
spacecraft isa platform
.
.
.
equipment isa object
weapon isa object
hand-weapon isa weapon
.
.
.
.
```

This segment tells us that platform is linked to object (isa object), equipment is linked to object (isa object), etc. Now if one is clever one says, "AH-HA"!!!, since spacecraft isa platform, if platform is selected at the beginning of a game, and yes is the answer given when the game asks if you want a more specific category, spacecraft should be one of the new subcategories that appear.(having played with startrek and being an expert-you know this is true!). Similarly, if weapon is choosen then hand-weapon is a subcategory.

This is not the most general organizational scheme. It is one aspect of semantic networks that we specialized for our games, but at some later date hope to be able to make more general again. Until that time situations occur that one has to work around. In all structured data, the structure is not always of the isa form (ie. the upper level is not always a class, of which the lower level is a form of). A simple example is a DOD database. The true structure is

```
Executive Department
  Agriculture
  Commerce
  Defense
    Navy
    Army
    Airforce
  HUD
  .
  .
  .
  .
```

Now, while Defense isa Executive Department seems correct, Navy isa Defence isn't right. The problem is of course that Department of Defense is not a class (something an object can belong to). Later a not too good method of using spelling and templates will be used to partially get around this problem.

Similarly, if one has a geography database and wants to consider increasingly smaller divisions, no nice way at present exists to set up this type of database. That is say the structure is:

```
California
  San-Diego-County
    La-Jolla
      Prospect-Road
        favorite-ice-cream-store
```

Again this isn't an isa structure. Someday we should be able to handle this. If this is your problem be creative.

Being familar with "objects", it is now time to meet some more of the players in the database. A simple one is attribute. An attribute is just a thing not connected to the isa tree. These are things that you don't want the player to select, but are interesting none the less. In the startrek database, such things as UFP, liquids, passengers have no isas, but appear:

```
Ptolomy origin UFP
mk-I cargo liquids
```

Next in difficulty to objects are relations. Relations are the verbs in the phrases we've been looking at. Setting up relations is a more difficult task, since object exists in the beginning, but one must create relations(including isa). Much of the hocus-pocus of the intital segment given in Section 1a is about setting up relations. What follows is a partial explaination of this manditory junk.

```
relation SET F_relation
isa      SET F_relation
isa      isa relation
```

These are the first three statements at the beginning of every database. They define the relations relation and isa. The SET (note the capitalization) sets flags on whatever is on the left hand side. In this case, the primitive (semnet) flag "F_relation" is set. So, the first statement states that relation is a relation. The second that isa is a relation, and the third that isa is a relation. This then starts up a relation tree, distinct from the object tree which contains the relations (verbs). In fact only those things that have the F_relation flag set can be used as the verb (middle word).

Possibly you are wondering why couldn't we have

```
                    relation SET F_relation    (****bad
                    isa relation relation          usage****)
```

This brings us to the world of inheritence. Flags carry or are inherited down the isa tree. So, if you set the F_relation flag on relation all those things xxxxx that appear in a phrase of the form

```
                    xxxxx isa relation
```

have the relation flag set. This being so, it seems highly unusual that another relation will need to use this primitive, just isa all your relations to relation or something that isa relation. Also note that isa is the only relation that allows inheritence of flags, so this explains why isa relation relation won't work, since isa still won't have the relation flag set.

The fourth line of the text contains the phrase:

```
                    same-as isa relation
```

After this declaration the system now knows you can have synomyns. Use is fairly straightforward.

```
                    Department-of-defense isa object
                    DOD  same-as Department-of-Defense
```

What synonyms allow the games to due is to take a synonym as a correct answer when the base word is required.

Next to understand is spelling. Those wonderful elves that brought you this system bring you spelling. As mentioned above, the games use the database to generate questions. How is this magical task accomplished? Through spelling templates!

Before you spell you must meet another semnet primitive. This is F_unique. What F_unique indicates is that given relation rrrrr and a subject sssss the phrase

```
                    sssss rrrrr xxxxx
```

is unique to xxxxx. There cannot be yyyyy such that 'sssss rrrrr yyyyy' is also true. A simple example is the relation sex (as an attribute not as a quantity). Obviously one wants this to be unique.

```
Person isa object
Fred isa Person
sex isa relation
sex SET F_unique
Fred sex male
```

In this way learn will object if at some later point in the text one has Fred sex female. Another example is isa. The sixth line of the standard heading is isa SET F_unique. The reason for this is simple. The isa tree must be a tree, not a network or a mesh or other messy thing. This way every child(subject) has only one parent(object), in the isa tree.

It is triangles (learned phrases) that are spelled. Spelling is accomplished by connecting templates to the relation of the triangle. This then gives us the class of template relations. There are presently two template relations used by the games. The spelling template relation gives relations templates for statements while the yes-no template relation provides templates for questions that can be answered by either yes or no. The defining phrases for these templates are

```
template isa relation
template SET F_unique
spelling isa template
yes-no isa template
```

Templates are statements inclosed in double quotes. Every instance of %s,%r,%o is replaced by the subject, the relation, or the object of the triangle being spelled. A simple example of this the spellings for isa:

```
isa spelling "%s is a type of %o"
isa yes-no   "Is %s a type of %o"
```

The quotes are removed by learn. Notice %r the relation symbol (which would be replaced by isa) is missing and instead is spelled out as "is a type of".

Now if we had a phrase Fred isa twit, and a game picked that phrase to spell out it would generate:

```
Fred is a type of twit
```

and at an appropriate point would ask:

```
Is Fred a type of twit
```

Similarly, in the startrek database we find:

```
inter-obj-relation isa relation
has isa inter-obj-relation
has yes-no "Does %s have %o"
shuttlecraft has impulse-drive
```

So at the appropriate point, we would expect the question

```
                                Does shuttlecraft have impulse-drive
```

Now it is becoming clear about the three types of relations mentioned in Section 1a. Looking at the header one sees

```
                                inter-obj-relation isa relation
                                inter-obj-relation SET F_leftObj
                                inter-obj-relation SET F_rightObj
                                inter-obj-relation IMPORTANCE 2
```

Making that great leap of intuition that seperates us from databases you conclude that F_leftObj and F_rightObj are semnet primitives and are some sort of flags. Correct. These flags guarantee that both the subject and the object are connected by the object tree to object at the top of the tree.(by now you have noticed that object has many uses. This is due to good terminology). This then explains the term inter-obj-relation. Relations of this type are between objects.

As mentioned before, flags are inherited down the isa tree. It is sometimes necessary to clear flags. Especially these two Obj flags. A case of this is the relation carries in the space database. Instead of setting up an obj-attribute/or obj relation (that is the object of the phrase could be either an attribute or an object), the following was done:

```
                                carries isa inter-obj-relation
                                carries CLEAR F_right Obj
```

which does what is needed, since it was decided that something could carry both things connected to the isa tree and things not connected. Similarly the relation attribute is set up for just those cases in which the object is an attribute.

```
                                attribute isa relation
                                attribute SET F_leftObj
                                .
                                .
                                .
                                origin isa attribute
                                .
                                .
                                .
                                Constitution origin UFP
```

In this way the various origins are not objects, and we have a class of relations called attributes which only require objects on the left.

Gosh, you say. Way back yonder you saw IMPORTANCE and asked if this was (I know it sounds silly) important? Of course!!! Importance is one of those 'under construction' areas of the Gaming System that right now is not as useful as we plan to make it. Importance is supposed to divide up the knowlege into various importance levels. Presently it just decides what is shown in certain games.

Our present policy is that structure should be of importance 1, database bookkeeping is of importance 0, and everything else is of importance 2. Structure is the isa's and relation. The only exception in the space database is origin which for some reason the author of the database decided should only be 1. If importance is 1, then flash will not ask any phrase having a relation of importance one, or inheriting an inportance one. It is important for the health of the games that everything that is considered user-visible should be of importance one or greater. This is IMPORTANT!!!!!

Some of you out there might wonder what database bookkeeping is all about. Well, wonder no more. Database bookkeeping is all those silly little things we put in the database to make our games work. For instance, in the startrek database you see that there is a triangle of the form 'hermes picture hermes2'. That is in the database so that the picquiz game can find out from the database that there is a picture of the hermes on the system disk. Note that, the way things are now, 'relation' is importance zero, so things that are relations must have their importance expicitly set to be non-zero or they will inherit their importance from 'relation'.

The last subject of interest is numbers. Numbers are represented in two distinct ways in the Gaming System. The first way is as a quantifier to a phrase.

```
Constitution has 4 personnel-transporter
Constitution has 5 emergency-transporter
Constitution has 2 cargo-transporter
```

The second way is to make the relation receive a quantity. This not suprisingly requires that the relation have a semnet primitive flag set. So accordingly in the startrek database

```
quantity isa relation
quantity SET F_quantity
.
.
.
obj-quantity isa quantity
.
.
.
length isa obj-quantity
width isa obj-quantity
.
.
.
Constitution length 289 meters
```

So, in this case you see that with a relation that can have a quantity on the right side, you also get free of charge the units.(though units are not required).

## SECTION 4
## LEARNING

My clients pay me to study the possibilities,
not the probabilities.
Perry Mason

At this point, you should have a wonderful text version of the database. Now all that is necessary to have the system accept this new batch of knowlege, is to 'learn up' the database (convert it to node form) and place it on the student disk, notifying the system that the new database is present.

The first aspect of the learn program that a novice notices is that it is not especially friendly to the user. This being so, be nice to it.

Place the administration disk in the Terak and reboot it. Place the disk with the text version of the database in the top drive (this drive is the #5 drive). The system will ask you to put a student disk in the system, respond by pushing the quit button (top left button). The system will come up. Now select the learn program. The learn program will respond with:

```
Learn-Semantic Network Database Constructor [ date ]
Command file(type CONSOLE: to use Keyboard)?
```

To which you respond CONSOLE:. (We have set up learn to use command files-but this is for later). The program now asks some questions on how you want learn to react to errors. The first is

```
Stop on errors encountered during learn(y/n)?
```

At this stage it is probably better to stop, correct the error and then relearn, so type y. If you had typed n the system then asks

```
Write errors to printer?(y/n)
```

If you type y and you have a printer attached, errors are written to the printer. A n causes the errors to be written to the screen.

The system then responds cryptically:

```
[NEW/OLD] database name
```

What it now wants is for example NEW OHWOW which tells it that we are making a NEW database whose name is OHWOW. As you can see, there is no way to specify which disk OHWOW will be placed on and in fact it goes to the bottom drive(the #4 drive). The system next responds with some random comments and then asks:

```
Input file:
```

To which you type in the name of the database text file. If the file is called FOO and is on device #5, you would respond #5:foo. At this point the system should take off. New triangles are printed at the bottom of the screen and a window into memory is at the top. Now sit back and watch your database being built.

After the file has been read in, the system asks:

```
input file:
```

If your database was large enough to be spread in more than one file type in the name of the next file. If this was the last file just type in return. A summary of the learn is then provided, with errors being listed in the file lrn.err.text.

If things don't work out:here are the errors and possible causes

unmatched quote

unmatched "("

more than 4 nodes in formula-possible two word node that wasn't quoted

illegal number-with four items on a line the system expects the third one to be a number and it wasn't

invalid flag number-flag must be in set {F_relation, F_unique,F_quantity,F_leftObj, F_rightObj} or an integer between 1 and 14

flag number out of range-flag number not between 1 and 14

invalid flag for Triange-triangle flags must be between 1 and 3

invalid importance-importance must be an integer

importance out of range-importance must be between 0 and 3

only 2 nodes in formula-must make triangles!!

The last task that remains is to install the new database on the student disk. There are two ways to do this. The first is to use the install program which may be available to you. If so just answer its questions and it will do all the work for you. If not just follow the following simple steps.

1.) Get into the filer.

2.) Transfer the three database files: dbname.names, dbname.nodes, dbname.index to the student disk. (note: for dbname you substitute the name of the database (as in OHWO.names for dbname.names)

3.) If you are just updating a databases you are done.

3a.) If this is a new database, you must now edit the file databases.text on the student disk. To this file you must add three lines. After the last line in the file add a blank line. Then add a line with dbname. Finally, add a line with a short description of the database. As an example: If databases.text originally contained:

```
SPACE
Startrek Database

STATES
U.S States-facts and figures
```

and you wanted to add the database OHWO just insert the following lines at the end of databases.text.

```
OHWO
Nothing Much of Interest
```

So the final file looks like:

```
SPACE
Startrek Database

STATES
U.S. States-facts and figures

OHWO
Nothing Much of Interest
```

Now you are ready to use the database. Just put in the student disk and the system disk and play away.

^0

## Auto Database

```
relation SET F_relation
isa SET F_relation
isa isa relation
same-as isa relation
isa SET F_unique
same-as IMPORTANCE 0
isa IMPORTANCE 1
quantity isa relation
quantity SET F_quantity
 quantity SET F_unique
 spelling isa relation
 relation spelling "%s %r %o"
 isa spelling "%s is a type of %o"

 obj-quantity isa quantity
 obj-quantity SET F_leftObj
 obj-quantity IMPORTANCE 2

 attribute isa relation
 attribute SET F_leftObj
 attribute IMPORTANCE 2

 inter-obj-relation isa relation
 inter-obj-relation SET F_leftObj
 inter-obj-relation SET F_rightObj
 inter-obj-relation  IMPORTANCE 2

 connects-to isa     inter-obj-relation

 connects-to spelling "%s connects to %o"

 functionally-connected-to isa      inter-obj-relation

 functionally-connected-to spelling "%s functionally connects to %o"

 constructed-with isa attribute
 constructed-with spelling "%s is constructed with %o"

 use isa attribute
 use spelling "%s's use is to %o"
 use IMPORTANCE 1

 has  isa  attribute

 mounts-on isa        inter-obj-relation

 mounts-on spelling "%s mounts on %o"

 part isa object

 steering-part isa part
 steering-gear isa steering-part
 steering-wheel isa steering-part
```

```
steering-column isa steering-part
steering-cable isa steering-part
tie-rod isa steering-part
spindle-arm isa steering-part
bellows isa steering-part

tie-rod connects-to bellows
tie-rod has threads
bellows connects-to steering-gear
bellows constructed-with no-metal
steering-gear connects-to steering-cable
steering-gear has yoke-cover
steering-cable connects-to steering-gear
steering-cable connects-to steering-wheel
steering-column connects-to steering-wheel
steering-wheel connects-to steering-cable
steering-wheel use "turn the car"
steering-column use "support the steering wheel"
steering-cable use "turn steering gear"
spindle-arm use "support the bearings"
bellows use "seal in lubricant"

engine-part isa part
piston isa engine-part
intake-manifold isa engine-part
connecting-rod isa engine-part
crankshaft isa engine-part
oil-pan isa engine-part
camshaft isa engine-part
oil-pump isa engine-part
cylinder-block isa engine-part
rocker-arm isa engine-part
radiator isa engine-part
raditor-fan isa engine-part
water-pump isa engine-part
thermostat isa engine-part
distributor isa engine-part
spark-plug isa engine-part
starter-motor isa engine-part
carburetor isa engine-part
air-filter isa engine-part
exhaust-manifold isa engine-part
battery isa engine-part
push-rod isa engine-part
muffler isa engine-part
exhaust-pipe isa engine-part
cylinder-head isa engine-part
valve isa engine-part
valve-guide isa engine-part
oil-pump-gear isa engine-part
distributor-gear isa engine-part

air-cleaner same-as air-filter
head same-as cylinder-head
crank same-as crankshaft
starter same-as starter-motor
block same-as cylinder-block
```

111.
112. cylinder-head use "house the camshaft"
113. exhaust-pipe use "expell unwanted engine gases"
114. muffler use "quiet engine noise"
115. radiator use "remove heat from coolant"
116. water-pump use "circulate coolant in engine"
117. thermostat use "switch water pump on and off"
118. piston use "compress the gases in the cylinder"
119. oil-pan use "catch oil at bottom of engine"
120. oil-pan use "store oil at bottom of cylinder block"
121. distributor use "time/order spark plug firing"
122. spark-plug use "provide ignition"
123. oil-pump use "circulate oil in engine"
124. carburetor use "control gas/air mixture"
125. exhaust-manifold use "collect post-firing discharges"
126. exhaust-manifold use "channel exhaust gases"
127. air-filter use "keep carburetor clean"
128. thermostat use "monitor water temperature"
129. cylinder-head connects-to exhaust-manifold
130. cylinder-head mounts-on cylinder-block
131. cylinder-head has rocker-arms
132. cylinder-head has valve
133. cylinder-head has valve-guide
134. thermostat mounts-on cylinder-head
135. muffler connects-to exhaust-pipe.
136. exhaust-pipe connects-to muffler
137. exhaust-manifold connects-to exhaust-pipe
138. exhaust-manifold connects-to cylinder-head
139. exhaust-manifold functionally-connected-to muffler
140. radiator functionally-connected-to water-pump
141. radiator has cap
142. water-pump functionally-connected-to thermostat
143. piston connects-to connecting-rod
144. piston has rings
145. connecting-rod connects-to crankshaft
146. crankshaft functionally-connected-to camshaft
147. camshaft connects-to push-rod
148. camshaft has sprocket-timing-gear
149. push-rod connects-to rocker-arm
150. oil-pan mounts-on cylinder-block
151. distributor mounts-on cylinder-head
152. carburetor mounts-on cylinder-head
153. battery functionally-connected-to starter-motor
154. spark-plug mounts-on cylinder-head
155. air-filter mounts-on carburetor
156. oil-pump mounts-on cylinder-block
157. air-filter connects-to carburetor
158. water-pump functionally-connected-to radiator
159. thermostat functionally-connected-to water-pump
160. connecting-rod connects-to piston
161. crankshaft connects-to connecting-rod
162. crankshaft has main-bearings
163. push-rod connects-to camshaft
164. rocker-arm connects-to push-rod
165. rocker-arm connects-to valve
166. valve connects-to rocker-arm
167. starter-motor connects-to battery

```
cylinder-head connects-to intake-manifold
carburetor connects-to intake-manifold
carburetor connects-to intake-manifold
distributor has points
distributor has cap
battery has terminals
oil-pump-gear mounts-on camshaft
distributor-gear mounts-on camshaft
camshaft connects-to crankshaft

body-part isa part
hood isa body-part
trunk isa body-part
frame isa body-part
door isa body-part
window isa body-part
sun-roof isa body-part
grill isa body-part

bonnet same_as hood
chassis same_as frame
moon-roof same-as sun-roof
door has armrests
door has locks
cylinder-block mounts-on frame
hood use "cover the engine"
trunk use "store spare tire"
trunk use "hold luggage"
frame use "support the car"
door use "allow easy interior access"
window use "allow difficult interior access"
window use "ventilate interior"
window use "increase visibility"
sun-roof use "let the sun shine in"
grill use "collect insects"

interior-part isa part
interior-frill isa interior-part
interior-standard isa interior-part
air-conditioneer isa interior-frill
radio isa interior-frill
cassette isa interior-frill
speaker isa interior-frill
fuzzy-dice isa interior-frill
fuzz-buster isa interior-frill
heater isa interior-standard
interior-light isa interior-standard
seat isa interior-standard
instrument-panel isa interior-standard
ashtray isa interior-standard
glove-compartment isa interior-standard
external-frill isa part
hub-cap isa external-frill
antenna isa external-frill

wheel-cover same-as hub-cap
map-light same-as interior-light
```

```
225.  cassete-deck same_as cassette
226.  bench same_as seat
227.  dash-board same-as instrument-panel
228.  radar-detector same-as fuzz-buster
229.  glove-box same-as glove-compartment
230.
231.  air-conditioner use "cool you in the summer"
232.  heater use "warm you in the winter"
233.  interior-light use "read maps at night"
234.  interior-light use "see the radio dial at night"
235.  cassette use "allow choice of sound "
236.  cassette use "play tapes"
237.  instrument-panel use "indicate car parameters"
238.  fuzzy-dice use "prove driver is a cool dude"
239.  fuzz-buster use "avoid speeding ticket"
240.  ashtray use "store loose change"
241.  glove-compartment use "store maps & miscellany"
242.  glove-compartment use "store gloves"
243.  antenna functionally-connected-to radio
244.  speaker functionally-connected-to radio
245.  cassette functionally-connected-to radio
246.  seat mounts-on frame
247.
248.  safety-part isa part
249.  horn isa safety-part
250.  brake isa satety-part
251.  head-light isa safety-part
252.  tail-light isa safety-part
253.  rearview-mirror isa safety-part
254.  windshield-wiper isa safety-part
255.  seat-belt isa safety-part
256.  air-bag isa safety-part
257.  bumper isa safety-part
258.  warning-light isa safety-part
259.  turn-signal isa safety-part
260.  side-mirror isa safety-part
261.
262.  brake-light same-as tail-light
263.  blinker same-as turn-signal
264.  safety-belt same-as seat-belt
265.  wiper same_as windshield-wiper
266.  idiot-light same_as warning-light
267.  fender same-as bumper
268.  turn-indicator same-as turn-signal
269.
270.  horn use "scare cyclists"
271.  horn use "announce your presence"
272.  brake use "slow you down"
273.  head-light use "increase forward visibility"
274.  tail-light use "warn drivers astern"
275.  rearview-mirror use "increase visibility"
276.  windshield-wiper use "allow driving in the rain"
277.  bumper use "keep minor accidents minor"
278.  seat-belt use "prevent ejection"
279.  seat-belt use "prevent injury"
280.  air-bag use "prevent injury"
281.  air-bag use "inflate on impact"
```

```
282.  warning-light use "indicate malfunction"
283.  turn-signal use "indicate direction change"
284.  side-mirror use "show adjacent lanes"
285.  side-mirror use "increase visibility"
286.  side-mirror mounts-on body
287.  rearview-mirror mounts-on windshield
288.  windshield-wiper has blade
289.  bumper mqunts-on frame
290.
291.  drive-train-part isa part
292.  u-joint isa drive-train-part
293.  rear-axle isa drive-train-part
294.  transmission isa drive-train-part
295.  driveshaft isa drive-train-part
296.  differential isa drive-train-part
297.  clutch isa drive-train-part
298.  wheel isa drive-train-part
299.  shift-lever isa drive-train-part
300.  tire isa drive-train-part
301.
302.  universal-joint same-as u-joint
303.  axle same-as rear-axle
304.  torque-tube same-as driveshaft
305.
306.  u-joint  mounts-on drive-shaft
307.  shift-lever functionally-connected-to transmission
308.  clutch functionally-connected-to transmission
309.  transmission connects-to driveshaft
310.  driveshaft connects-to differential
311.  differential mounts-on rear-axle
312.  rear-axle connects-to wheel
313.  tire mounts-on wheel
314.  transmission mounts-on frame
315.  u-joint use "allow for transmission misalignment"
316.  u-joint use "connect the driveshaft and differential"
317.  rear-axle use "transmit power to rear wheels"
318.  transmission use "keep engine in proper RPM range"
319.  shift-lever use "allow gear change"
320.
321.  suspension-part isa part
322.  ball-joint isa suspension-part
323.  shock-absorber isa suspension-part
324.  control-arm isa suspension-part
325.  spindle isa suspension-part
326.  stabilizer-bar isa suspension-part
327.  coil-spring isa suspension-part
328.  leaf-spring isa suspension-part
329.  hanger isa suspension-part
330.
331.  torsion-bar same-as stabilizer-bar
332.  control-arm connects-to ball-joint
333.  spindle connects-to ball-joint
334.  coil-spring mounts-on control-arm
335.  stabilizer-bar connects-to control-arm
336.  control-arm mounts-on frame
337.  leaf-spring connects-to hanger
338.  hanger connects-to frame
```

339. shock-absorber use "absorb the bumps"
340.

^O

## Where the Databases Live

Databases have two forms, a text form and a form the games understand. If you use the filer on a student disk you will note that for a given database name, say Cars, you will see files

```
Cars.text
Cars.nodes
Cars.names
Cars.index
```

The last three files are necessary for the games. For the games to know that these files exist, an entry must be made in the file,

```
databases.text
```

If a new database has just been made, by either dbedit or learn, then the .nodes, .name, .index file must be transfered to the student disk. After this has been done, edit databases.text. The file should be in the form of

```
database name 1
database description 1
database name 2
database description 2
```

and so forth.

A typical system may have:

```
UCsoviet
The Soviet Navy - From Janes
Cities
Cities of the U.S.
```

In this file just add the following two lines at the bottom:

```
Cars
Cars of the World
```

which tells the games about the new database.

## Dumper

The dumper program converts database files to text files. It is in some sense the inverse operation to the learn program. The text files provide backup for the datbases. It is also necessary to dump files once they have been edited by dbedit to provide for more compact databases.

To operate the dumper, select it on the Adm_5.2 disk. The dumper comes up with a list of databases, from the student disk. If the databse is not on the student disk, see the document, "Where the Databases Live", to find out how to install it.

Select the database to be dumped, by pressing the SELECT key when the cursor is in front of the appropriate database. The program will respond with:

```
write to #5:'name of database'.text[YES/NO]
```

where 'name of database' is the name of the selected database. If this is ok press the YES button. If another name is to be used, press NO and then the system will ask:

```
File to output to:
```

to which the name of the file is typed in. Note that #5: must prefix the name if it is to go on the top disk.

At this point the program will happily generate the text file, with no further prompting.

## Creating a Database Using DBedit

This document is a tutorial in creating a database using the program dbedit. The technique is lead the reader through the birth of an example database, pointing out construction hints as we go. The database that will be created is that of

States of the United States

To construct a new database, select dbedit from the adminsitration disk. Dbedit comes up and asks if a new database is being constructed. Press the YES key. Dbedit then asks for the database name.

States

is typed in. The system then asks:

Top Drive[YES/NO]

allowing the option of constructing the database on the botton drive. In almost all cases, the database is constructed on the top disk, so press YES. After a brief pause as the files are set up, two menus appear on the screen.

The menu on the left displays categories, while the screen on the right displays information about those categories. At this point, in the Categories menu is the word object, and in the Description menu, the description of object of which there isn't one yet. Under object we will put all the "things" of the database. The first category under object to add is state. Press the 'a' key.

The prompt line at the bottom of the screen then requests:

Enter new item:

Just type in:

State

In a few seconds the menu now contains object and indented below it the word state. It is now time to enter all the states. To do this we must first enter the category State. Move the cursor ( the > ) next to State using the down arrow key and press the SELECT key. Now State is alone in the Categories box. Press the BACK key. The intial arrangement of object and state reappear. By using the SELECT and BACK keys, you can move into subcategories (like state) or back up to higher categories (like object).

Again move the cursor in front of state and press the SELECT key. The states are then added under state. Press the 'a' key. Again the prompt

Enter new item:

appears. Type in Alabama. It then appears under state. In this way, all the states can be added. The menu should begin to look like:

```
State
  Alabama
  Alaska
  Arizona
```

and so forth. Now while typing in I got to Hawaii and looked at the menu:

```
Hawaii
Georgia
ii
Deleware
Florida
Colorado
Connecticut
Arkansas
alifornia
Alabama
Alaska
Arizona
```

There are at least two mistakes in this list. ii is certainly not a state and while most Califorians are missing a few marbles, it isn't nice to leave off the first letter of their state. To get rid of the ii, just move the cursor in front of the ii and press 'r' for remove. Remove asks for confirmation. Press the YES key. It then asks if subtrees should be deleted. More will be said about that later, but for now just press the YES key. The ii is now gone.

Move the cursor in front of aliforina. We want to rename it, so press the N key for Name Options. The promt states:

```
R: Rename, S: Synonym
QUIT to exit
```

Press 'r' to rename. The prompt asks:

```
Rename alifornia to be:
```

Just type in California. Notice on the the menu California has replaced alifornia

It is a good idea to save the database every so often in case a system crash, or power failure causes you to loose everything. Just press the QUIT key. Confirm the quit by pressing YES. If this is a new database, you must notify the system that it can now use the database. See the document, "Where the Databases Live". To redbedit the file run dbedit again. This time when the system asks if you wish to create a new database press the NO key. A menu will appear. If you put the new database name correctly in the file databases.text, its description will appear in the menu. Just select it and continue working on the database.

Let's next add the capitals of all the states. First we must add the object city. Using the BACK key get back to object. The menu will now contain object and state. Press the 'a' key and enter city. Now city is an object. Now under city we can start entering the capitals. Move the cursor in front of city and press SELECT. Now we are in the category city.

To enter the capitals, just press 'a' and add them. After the capitals are added, they must be hooked to their appropriate states. The way to do this is to add a new relation called capital. Press the 'e' key. The prompt asks

```
Enter category:
```

Type in 'relation'. The following relations are built in:

```
attribute
quantity
inter-obj-relation
isa
same-as
spelling
```

The type of relation you want is the inter-obj-relation. An inter-obj-relation is a relationship between two things that under object. In this case, we will want to be saying:

```
Minnesota's capital is St. Paul
```

Minnesota is under state which is under object, and St. Paul is under city which is under object, so capital will be an inter-obj-relaion. A type of relation not of the inter-obj-relation type is the attribute type of relation. In this type of relation the thing on the right (St. Paul in the above example is the thing on the right) doesn't have to be under object. If for instance we had a database of cars, then we might want the relation color. Now there is no reason why the various colors (green indigo etc) need be under object, so we could have the relation color be an attribute type relation.

In this case, capital is an inter-obj-relation, so move the cursor down to inter-obj-relation and press SELECT to get in the category inter-obj-relation. Press 'a' and add capital. We must also add a spelling for capital, so the games will know how to use it. Select capital. In the description menu notice:

```
capital isa inter-obj-relation
```

Press the right arrow key -->. You can now add descriptions of capital. Press the 'a' key. The prompt states:

```
Enter info: capital
```

The system is expecting you to enter a relation and something that fits the relation. The relation we want is spelling. So enter the following cryptic mess:

spelling "%s's capital is %o" Notice in the menu on the right, the following are present:

```
capital isa inter-obj-relation
capital spelling %s's capital is %o
```

The spelling is important, because this is the way that the games know how the lines should be output. With capital's spelling, Flash will ask the question:

```
What state's capital is Juneau
```

Without the spelling, it would have asked:

```
What state capital Juneau
```

Similarly, if a correct match has been made in Concentration, between Juneau and Alaska, the system responds,

```
Correct! Alaska's capital is Juneau
```

instead of:

```
Correct! Alaska capital Juneau
```

It should be clear now, that the games replace the %s in the spelling with the subject of the statement:

```
Alaska capital Juneau
```

which is Alaska, and the %o is replaced with Juneau, the object of the statement. Every relation should have a spelling to help the games do this nice output work.

You must also add the importance of this relation. Presently the only way to do this is to press the 'a' again and type

```
Enter info:capital IMPORTANCE 2
```

where you have typed in IMPORTANCE 2 to the prompt. Notice that IMPORTANCE must be capitalized. Soon you won't have to go through this:

Maybe...... Now use the left key <-- to go back to the categories box. Press 'e' to enter a new category and then type in state to get to the states. Now using the down arrow key move the cursor in front of Alaska. Press SELECT. In the description box appears:

```
Alaska isa state
```

Note the heading for the left menu is Alaska and it will contain all the information the system knows about this state. We want to add a description to Alaska, that is we want to tell the system that Alaska's capital is Juneau. This is just like telling the system the spelling of capital. Use the right arrow key to move into Alaska's description box. Press 'a'. The prompt states:

```
Enter info: Alaska
```

Type in

capital Juneau

In the description menu appears:

, Alaska capital Juneau

As I was adding the cities, I came accross St. Paul. Now it would be nice if the system recognized Saint Paul also, and there is a way to do that. Position the cursor in front of St. Paul and press 'n' for name options. Now press 's' for synonym. The prompt asks:

Name same-as St. Paul: Input Name:

to which you type in Saint Paul.

All that remains is to describe what type of relation to make your relations and then the wonderful world of database construction is at hand. Attribute type relations are those in which the object, thing on the right isn't important enough to be under object. In the states database this would be like the relation nicname:

Ohio nicname "Buckeye State"
Texas nicname "Longhorn State"

in which Buckeye State and Longhorn State certainly aren't too important, so enter the attribute category, by pressing 'e' key while in the Categories Menu and typing attribute. Then press 'a' to add nicname as an attribute relation.

The last relation of interest is the quantity relation. If it is necessary to note the number of an item such as the number of wives of a Brittish king:

HenryVII married 7 women

for example. Another example is

"United States" has 50 states

If the quantity is between 0 and 16383 ( the quantity being 7 and 50 in the two examples above) then nothing special is necessary. If the quantity is out of that range, then it is necessary to make a special type of relation. A case where this is necessary is:

Nitrogen freezes -209 Fahrenheit

In this case it would be necessary to define the relation freezes as a quantity type relation. This type of relation allows positive and negative numbers up to 37 digits long, with six of the digits significant.

So that's it. Get to work.

^0

# User's Manual-Dbedit

The dbedit program provides an alternate way of making and edititng a database. The editor provides full facility for modifying adatabase. Once the databse is made, the documents, "Dumper", "Using Learn", and "Where the Databases Live" provide instructins for dumping the database to textfile, relearning it to dbform and for installing new databases on the student disk. Since the editor does not presently do memory management, to get a more compact database it is necessary to dump and relearn databses after modifications have been made.

a) Creating a Database

Creating a new database does not differ radically from editting a database. The major difference is a conceptual one. When editting a database a framework already exists for making changes. A new database provides untold opportunity for creativity and messing up. To cover that level of the problem see the document "Creating a Database Using Dbedit". Here we cover some of the different mechanics from editing a database.

At the beginning of dbedit the system asks:

```
Create a new database [YES/NO]
```

to which YES must be pressed. The system then asks:

```
Top Drive [YES/NO]
```

which gives the opportunity of placing the database on the lower drive. This is almost never done, so press YES.

After a short pause, the dbedit main display appears.

On the left is a menu with the objects of the database. Initially, only object is present. Also if E is pressed and then 'relation' is typed in at the prompt, a menu containing the built-in relations:

```
attribute
quantity
inter-obj-relation
isa
same-as
spelling
```

Type E again and type 'object' at the prompt, and below we will discuss the various options from this level.

b) Editing- from the Categories Menu

Introduction:
Listed below are the commands available from the category menu. It is in this menu that editing on the objects of the database can be done. Adding new objects, moving objects, deleting objects, renaming objects (such as categories and subcategories) are available options

here. To enter the Description Menu press the right arrow key, and to enter the template search mode press t.

SELECT: This command causes the item in which the cursor is in front of to become the category of interest. The item is moved to the head of the Categories Menu and indented below it, its subcategories are displayed.

D : This command causes the item in which the cursor is in front of to be described. All the relations in which the item participates in appear in the description box.

BACK : This command causes the supercategory of the head object in the Category menu to become the head object.

E : This command causes the system to inquire if a new category is to be entered. The prompt asks:

```
Enter category:
```

to which the new category is typed in followed by RETURN. If the category is not know to the database the error:

```
name not found
```

occurs and the original category is retained.

A : This command causes the system to add a new subcategory to the head category. The prompt asks:

```
Enter new item:
```

to which the new item is typed in followed by RETURN. Be careful of spelling. If the item is already known to the system the error:

```
That name is already in use
```

occurs and no operation is performed.

R : This command causes an item to be removed from the database. The prompt asks:

```
Confirm Delete[YES/NO]
```

and if YES is pressed asks:

```
Delete Subtrees[YES/NO]
```

which if answered YES will delete all subcategories of the item. If NO is pressed the subtrees will be inherited by the supercategory of the deleted item.

M : This commmand moves the item that the cursor is in front of to a different supercategory. Relations and the item object can not be moved.

N : The command allows access to the name option functions. These functions are R : for rename and S : for synonym. If after N is pressed R is pressed the prompt asks:

Rename oldname to be:

(oldname is the item to be renamed) to which the new name for the highlighted item is typed in followed by return. If the new name is already known to the system an error:

That name already in use

will occur and the name options mode will be aborted. If the S is pressed after the N has been pressed the prompt will ask:

Name same-as oldname Input Name:

(oldname is the item that is being synonymed) to which the synonym for the given object is typed in followed by return. Again if the synonym is already known the error:

That name already in use

will occur and the name options mode will be aborted.

ARROWS : These commands move the cursor in front of items (the up and down arrows) and accross to the Description Menu (the right arrow).

QUIT : This command causes the prompt:

Confirm Quit[YES/NO]

If YES is pressed  the editing session is over.  If NO is pressed no harm has been done. If YES is pressed and this is reediting of an existing database the prompt asks:

Update olddbname[YES/NO]

if YES is pressed the changes from this editing session are incorrporated in the old database. If NO is pressed the original database with no changes is kept.

T : This command causes the template search mode to be entered.

b) Editing- from the Description Menu

Introduction:

The Description Menus contains descriptions of items in the database. It is from this menu that descriptions can be added and deleted. To go to the Categories Menu press the left arrow keys. To enter the template search mode press T.

SELECT : This command shows the item in the triangle in which the cursor is in front of which is not presently being described.

E : This command causes an item to be described. The prompt asks:

Enter item:

to which the item to be described it typed in followed by RETURN. The Description Menu then fills with the description of the item.

A : This command allows a new description to be entered about the item being described. The prompt asks:

Enter info: item

(where item is being described) To which a relation followed by an object must be entered. If either of these items is more than a single word, it must be hyphenated or put in quotes. After these are typed in press RETURN. Numerous errors are possible here. See the appendix for explainations.

R : This command causes a description to be removed. The prompt asks:

Confirm Delete[YES/NO]

if YES is pressed the item is removed. If NO is pressed no harm is done.

b) Editing- from the Description Menu

Introduction:

The template search mode allows searching for triangles. The left menu is the search menu. In this menu, part of the triangle is specified and the system searches for all matches. The item any can be chosen, which matches any item. To return to the other menus, move the cursor in front of Return to Categories in the search menu and press SELECT.

This is the only method to discover what triangles relations participate in.

The user presses SELECT in front ~~of~~ either SUBJECT, OBJECT, or RELATION and then types in either the specific item for that category to be searched or the word any, which will match any item. The user can also specify chaining, importance, and whether all the matches should be retrieved.

Chaining causes the lowest items in the specified category to be matched. If SUBJECT = birds, RELATION = nest, OBJECT = caves and chaining = no then the system would try to find a triangle:

birds nest caves

which wouldn't have any matches. If chaining = yes, then the system would look at types of birds checking;

```
              bluejays nest caves
              robins nest caves
              starlings nest caves
                    etc.
```

importance specifies the minimum importance that are returned. 0 is good for almost all cases.

All matches = no can be used if just an example is wanted of that relation. It speeds things up.