Lesson 7 Lab 10.2 – File Access and Flowcharts

Critical Review

Outputting to a File using Raptor

The Output symbol is used to output data to a text file. When an Output symbol is reached during Raptor program execution, the system determines whether output has been *redirected.* If output has been *redirected,* meaning an output file has been specified, the output is written to the specified file. If output has not been redirected, it goes to the Master Console.

One version of redirecting output to a file is by creating a call symbol and adding the following:

```
Redirect_Output("file.txt")
```

Note: If the file specified already exists, it will be overwritten with no warning! All the file's previous contents will be lost!

The second version of Redirect_Output redirects output with a simple yes or true argument:

```
Redirect_Output(True)
```

This delays the selection of the output file to run time. When the Call symbol containing Redirect_Output is executed, a file selection dialog box will open, and the user can specify which file is to be used for output.

After a successful call to Redirect_Output, the program writes its output to the specified file. To reset Raptor so that subsequent Output symbols write their output to the Master Console, another call to Redirect_Output is used, this time with a False (No) argument:

```
Redirect_Output(False)
```

After this call is executed, the output file is closed, and subsequent outputs will again go to the Master Console.

Input to a File using Raptor

This is done the same way, except `Redirect_Input( )` is called.

To pull something in from a file, the input symbols are used.

This lab requires you to create a RAPTOR flowchart for the blood drive program in Lab 10.1.

Step 1: Start Raptor and open your *Lab 9-3*. Save this file as *Lab 10-2*. The *.rap* file extension will be added automatically.

Step 2: Remove the references no longer need as follows. Delete the `highPints` and `lowPints` variables, and then delete the `getHigh`, `getLow`, and `displayInfo` modules. With the modules, first delete the function calls, and then right click on the tabs and select Delete Subchart.

Step 3: In main after the module call to `getAverage`, add a call to `writeToFile`.

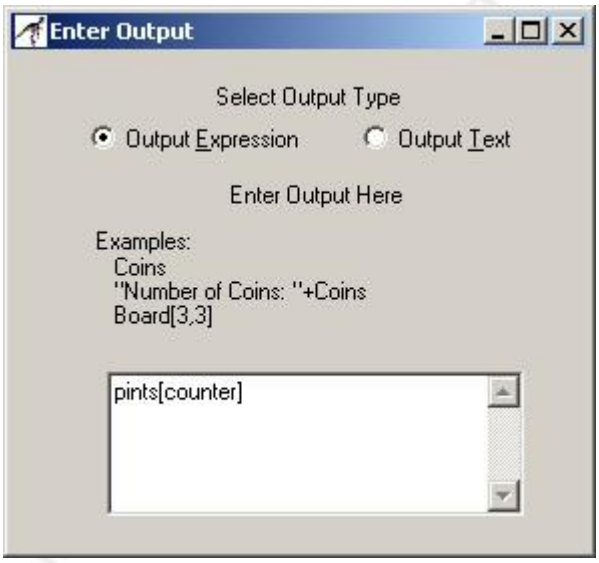
Step 4: Go to that module and add a call symbol. Add the following:
`Redirect_Output("blood1.txt").`

Step 5: Add an output symbol that prints the string `"Pints Each Hour".`

Step 6: Add an assignment symbol that sets `counter` to 1.

Step 7: Add a loop symbol that has the condition of `counter > 7`.

Step 8: If it is `False`, add an output symbol that prints `pints[counter]` to the file. This should look as follows:

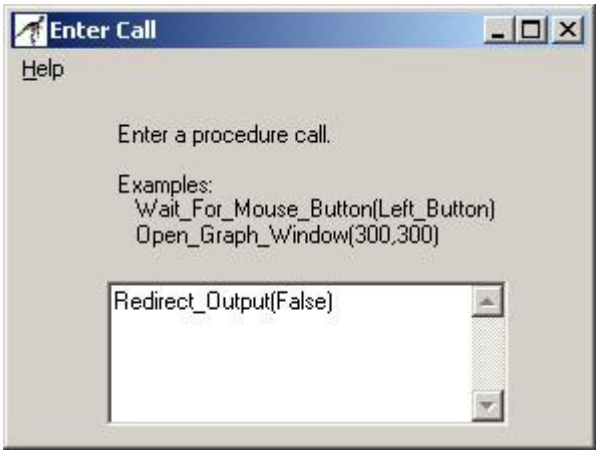


Step 9: Add an assignment statement that increments `counter` by 1.

Step 10: If it is `True`, add an output symbol that prints the string `"Average Pints"` to the file.

Step 11: Add an output symbol that prints `averagePints` to the file.

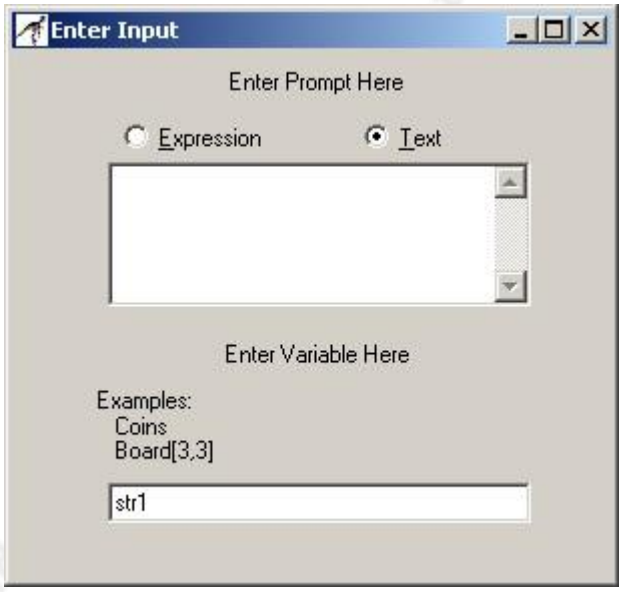
Step 12: Add a call symbol that closes the file. This should look as follows:



Step 13: In main after the call to `writeToFile`, add a call to `readFromFile`.

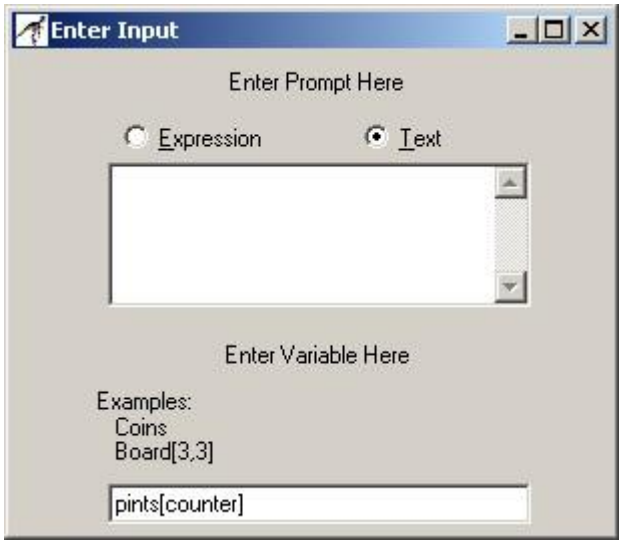
Step 14: In the `readFromFile` module, add a call symbol to `Redirect_Input`, such as `Redirect_Input("blood1.txt")`.

Step 15: Add an Input symbol that gets `str1`. This should look as follows:



Step 16: Add an assignment statement that sets `counter` to 1.

Step 17: Add a loop statement. If the loop is `False`, get the next value from the file and store it in `pints[counter]`. This should look as follows:



Step 18: Increment `counter` by 1.

Step 19: If the loop is `True`, get `str2` with an input symbol.

Step 20: Add an input symbol that gets `averagePints`.

Step 21: Add a call symbol that sets `Redirect_Input` to `False`.

Step 22: In the Main module, add an input symbol under the loop symbol. This should ask the user to enter 1 if they want to take in data and add to the file or 2 if they want to print information from the file. Store this in a variable called `option`.

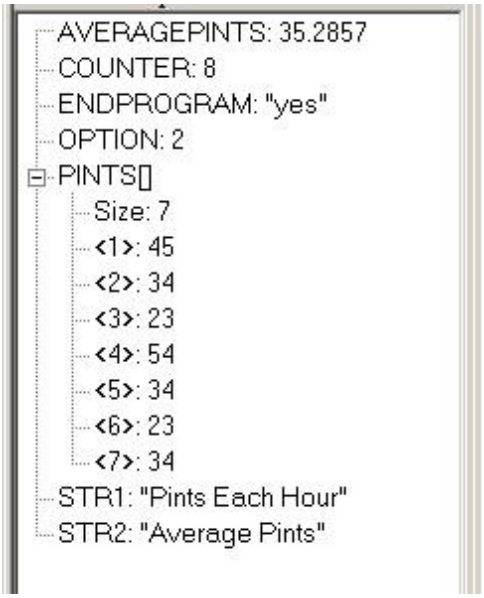
Step 23: Add a decision symbol that asks if `option` is equal to 1. If it is, call the `getPints`, `getTotal`, `getAverage`, and `writeToFile` module. If it is not, call the `readFromFile` module.

Step 24: Run your program once and be sure to select option 1 on the first time. This will create a file called `blood1.txt` in your directory where your Raptor flowchart is located. An example file might contain the following:

```
Pints Each Hour
45
34
23
54
34
23
34
Average Pints
35.2857
```

Step 25: Go to your file called `blood1.txt` and examine the contents. Paste the contents into a Comment box on the Main module of your flowchart.

Step 26: Run your program again, but select option 2. You can test to see if it is reading values properly into your program by examining the contents of the variables that are listed on the left. The following is an example:



Step 27: Execute the program to make sure it works and upload your completed RAPTOR (.rap) file as your submission.