Lab 1: Input, Processing, and Output

This lab accompanies Chapter 2 of *Starting Out with Programming Logic & Design*.

Name: Ulric Aird

Lab 1.1 – Algorithms

Critical Review

An algorithm is a set of well-designed logical steps that must take place in order to solve a problem.

The flow the algorithm takes is sequential. For example, before you process calculations, all data needed should be retrieved.

This lab requires you to think about the steps that take place in a program by writing algorithms. Read the following program prior to completing the lab.

```
Write a program that will take in basic information from a
student, including student name, degree name, number of credits
taken so far, and the total number of credits required in the
degree program.  The program will then calculate how many credits
are needed to graduate.  Display should include the student name,
the degree name, and credits left to graduate.
```

Step 1: Examine the following algorithm.

```
1. Get the student name.
2. Get the degree program name.
3. Subtract the number of credits taken so far from the required
   credits for the degree.
4. Get the number of credits required for the degree program.
5. Get the number of credits the student has taken so far.
6. Display the input information in Step 1 and 2.
7. Display the calculated information.
```

Step 2: What logic error do you spot and how would you fix it?

The error is we are subtracting before getting the credits required and credits taken

Solution: Input Data before subtracting

Step 3: What steps require user interaction (Ex: user must type in some input)?

Step 1, 2, 4 and 5

Lab 1.2 – Pseudocode

Critical Review

Pseudocode is an informal language that has no syntax rules and is not meant to be compiled or executed.

The flow the program takes is sequential. For example, before you ask for input, you should display what information you want from the user.

```
// Comments are done by putting two forward slashes
// before the lines you want to document. Comments
// are used to explain code.
```

Variables are named storage locations.

`Declare` is the keyword used before naming a variable. Data types are: `Real` for decimal numbers, `Integer` for whole numbers, and `String` for a series of characters.

Follow the rules for naming variables: (1) must be one word, no spaces, (2) usually no punctuation characters, only letters and numbers, and (3) name cannot start with a number.

`Display` is the keyword used to print something to the screen. Any information needed to be displayed to the user should be put inside quotation marks such as:

```
Display "This is how you print something to the screen"
```

When using display to print both a string and the value of a variable, a comma is used, such as:

```
Display "Here is the average: ", average.
```

`Input` is the keyword used to get the user to enter data. The data value entered by the user will be placed in the variable that follows the keyword input such as *Input variableName*.

`Set` is the keyword used before a calculation. Standard math operators are used, such as `+ - * / MOD ^`. Operators can be combined in one calculation, but it is wise to group expressions together using parentheses. Remember the order of operations. Some examples are:

```
Set sale = price - discount
Set average = (test1 + test2 + test3) / 3
```

This lab requires you to think about the steps that take place in a program by writing pseudocode. Read the following program prior to completing the lab.

```
Write a program that will take in basic information from a
student, including student name, degree name, number of credits
taken so far, and the total number of credits required in the
degree program.  The program will then calculate how many credits
are needed to graduate.  Display should include the student name,
the degree name, and credits left to graduate.
```

Step 1: This program is most easily solved using just five variables. Identify potential problems with the following variables declared in the pseudocode. Assume that the college has the ability to offer half credits. (Reference: Variable Names, page 39-40).

Variable Name	**Problem (Yes or No)**	**If Yes, what's wrong?**
`Declare Real creditsTaken`	No	
`Declare Real credits Degree`	Yes	There is a space thats not suppose to be there
`Declare Int creditsLeft`	No	
`Declare Real studentName`	Yes	it suppose to have a String data type, not a real
`Declare String degreeName`	No	

Step 2: Complete the pseudocode by writing the two missing lines. (Reference: Prompting the User, page 42).

```
Display "Enter student name."
Input studentName
Display "Enter degree program."
Input degreeName
Display "Enter Degree Credits. "
Input creditsDegree
Display "Enter the number of credits taken so far."
Input creditsTaken
```

Step 3: What two things are wrong with the following calculation? (Reference: Variable Assignment and Calculations, page 43).

```
creditsLeft = creditsTaken - creditsDegree
```

1. "creditsDegree" must be subtracted from "creditsTaken" not the other way around.

Step 4: Write the exact output you would expect from the following line of code if the user of the program enters "Bill Jones". (Reference: Displaying Multiple Items, page 40 – 41).

```
Display "The student's name is ", studentName
```

The student's name is Bill Jones

Step 5: Write the exact output you would expect from the following line of code if the user of the program enters a degree that is 63 credits in total and they have taken 40 credits. (Reference: Displaying Multiple Items, page 40 – 41).

```
Display "This program requires ", creditsDegree, " credits and
they have taken ",  creditsTaken, " so far."
```

Step 6: Complete the following pseudocode to solve the programming problem.

```
1. //This program takes in student information and calculates
2. //how many credits the student has left before graduation.
3. //Information is then printed to the screen.

4. //Declare variables
5. Declare Real creditsTaken
6. Declare Real creditsDegree
7. Declare Int creditsLeft
8. Declare String studentName
9. Declare String degreeName

10. //Ask for user input
11. Display "Enter student name."
12. Input studentName
13. Display "Degree Program. "
14. Input degreeName
15. Display "Enter degree credits."
16. Input creditsDegree
17. Display "Enter the number of credits taken so far."
18. Input creditsTaken

19. //Calculate remaining credits
20. creditsLeft = creditsDegree - creditsTaken

21. //Display student name, degree program, and credits left.
22. Display "The student's name is ", studentName
23. Display "Degree Program is ", degreeName
24. Display "Credits Remaining is ", creditsLeft
```

Lab 1.3 – Flowcharts

Critical Review

A flowchart is a diagram that graphically depicts the steps that take place in a program. Symbols are used to depict the various steps that need to happen within a program. Flow lines are used between the symbols to indicate the flow of the program.

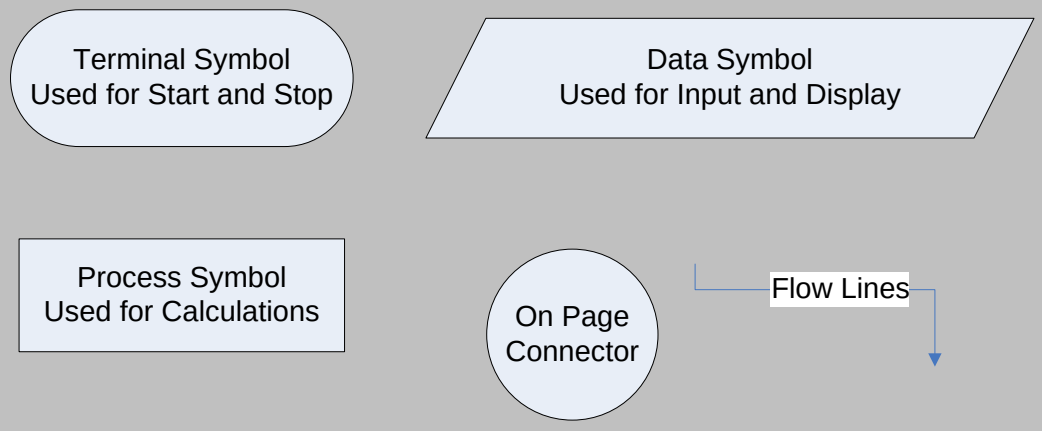
Ovals are used as terminal symbols, which indicate a start and stop to a program.

Parallelograms, the data symbol, are used for input and display statements.

Rectangles, the process symbol, are used for calculations and variable declarations.

On page connectors are used to link a flowchart that continues on the same page. The connecting system starts with the letter A, whereas A would appear in the two connectors that show the flow.

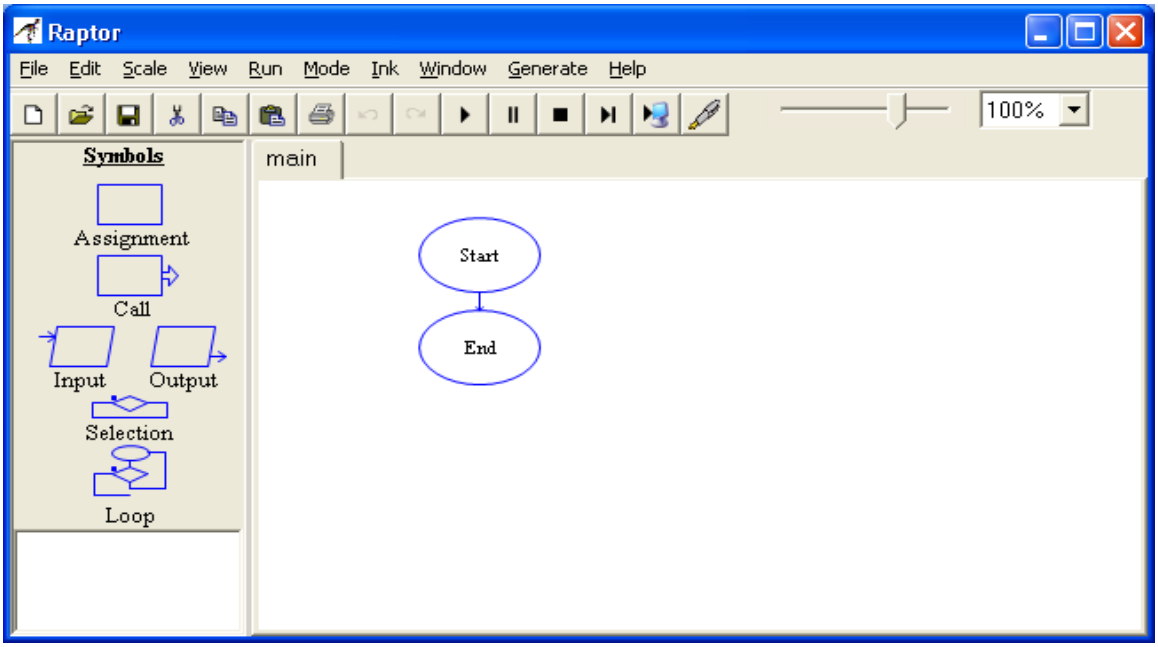
The statements inside the data and the process symbols can be written similarly to the statements used in pseudocode.



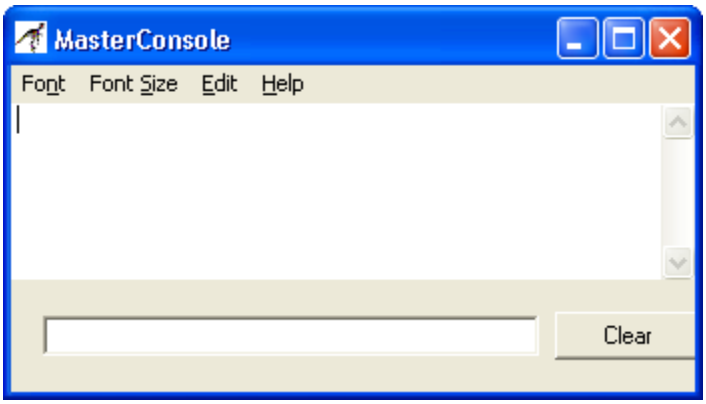
This lab requires you to think about the steps that take place in a program by designing a flowchart. While designing flowcharts can be done with paper and pencil, one mistake often requires a lot of erasing. Therefore, a flowcharting application such as Raptor or Visio should be used. This lab will give you a brief overview of Raptor. Read the following program prior to completing the lab.

```
Write a program that will take in basic information from a
student, including student name, degree name, number of credits
taken so far, and the total number of credits required in the
degree program.  The program will then calculate how many credits
are needed to graduate.  Display should include the student name,
the degree name, and credits left to graduate.
```

Step 1: Start Raptor; notice the Raptor screen. This window is your primary tool for creating a flowchart. Prior to adding symbols, save your document by clicking on File and then Save. Select your location and save the file as *Lab 1-3*. The *.rap* file extension will be added automatically.

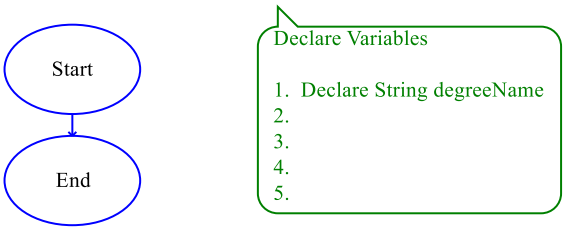


Step 2: Notice the MasterConsole screen. This window is used to show your program output once your flowchart is completed. The Clear button will clear the console to view a fresh run of your program.

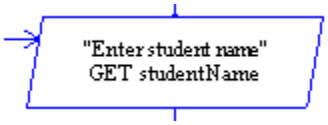


Step 3: Return to the Raptor screen to begin adding symbols into your flowchart. Your flowchart should follow the pseudocode in Lab 1-2, Step 6. While a rectangle is normally used

for declaring variables, there is no easy way to do this in Raptor. Since this is an important part of flowcharting, we will do this using a comment box. To do this, Right-Click on the Start symbol and select Comment. In the Enter Comment box, type the variables your program will need. Below is a start to how it should look.

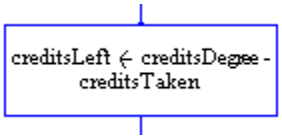


Step 4: The next step in your flowchart should be to ask for user input. Click the Input Symbol on the Left and Drag and Drop to the flow line between Start and Stop. Double Click on the Input Symbol to begin entering information. Enter `Enter student name` in the top box. Enter `studentName` in the variable box. Below is how it should look.

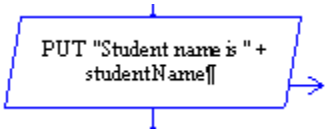


Step 5: Continue the Step 4 directions for all your input statements, changing each Input symbol to reflect the appropriate user interaction.

Step 6: The next step in your flowchart is to process any calculations that exist. Click on the Assignment symbol and drag it to the flow line between the last input statement and the end symbol. Double click on the Assignment symbol to enter your code. In the Set box, put the name of your storage variable. In the To box, put the expression part of your formula. Below is how it should look.



Step 7: The next step in your flowchart is to display the requested output to the screen. Click the Output symbol and drag it to the flow line between the assignment statement and the end symbol. Double click on the Output symbol to enter your code. Under Output Type, select Output Expression since we want to display both a sentence and the contents of a variable. In the box, type `"Student name is " + studentName`. Below is how it should look once you click Done.



Step 8: Continue the Step 7 directions for all your output statements, changing each Output symbol to reflect the appropriate requested output information.

Step 9: Once your flowchart is complete, click on Run and then Execute to Completion on the Raptor menu. Follow the flow of your program to see if it processes properly. Your Master Console window should show output similar to

```
Student name is Bill Jones
The degree program is Computer Programming
Credits left to graduation is 39
----Run finished----
```

Step 10: The final step is to insert your finished flowchart in the space below. Inside Raptor, select File and the Print to Clipboard from the menu. Inside Word in the space below, select Edit and Paste.

PASTE FLOWCHART HERE