CS262, Lab Assignment 3:
Branching, Loops and Functions
Due: Sunday, Feb 27 at 11:59 pm ET

Description:
The purpose of this assignment is to practice writing code using functions and control statements. You will code a C program that displays a menu to prompt the user for input. The program will display a menu to prompt the user to choose a "geometric shape" to be printed on the screen.

Your code must contain at least one of all the following control types:

- a `for()` loop
- a `while()` or a `do-while()` loop
- a `switch()` statement
- an `if-else` statement

The first thing your program will do is print a menu with these choices:

Menu Choice	Input Choices
Enter/Change Character	'C' or 'c'
Enter/Change Number	'N' or 'n'
Draw Line	'L' or 'l'
Draw Square	'S' or 's'
Draw Rectangle	'R' or 'r'
Draw Triangle (Left Justified)	'T' or 't'
Quit Program	'Q' or 'q'

A prompt is presented to the user to enter a choice. If the user enters a choice that is not a valid input, the message "The choice is invalid" is displayed and the menu is displayed again.

Your program must have at least five functions (besides `main()`) including:

- **A function that prints the menu** of choices for the user, prompts the user to enter a choice, and retrieves that choice.
- **A function that prompts the user to enter a single character**. The return value of the function be a `char` and will return the character value entered by the user. This return value will be stored in a local variable, `C`, in `main()`.

  The initial default value of this `C` will be ' ' (*blank character*).
- **A function that prompts the user to enter a numerical value** between 1 and 15 (inclusive). If the user enters a value outside this range, the user is prompted to re-enter a value until a proper value is entered. The return value of the function be an `int` and will return the value entered by the user. This return value will be stored in a local variable, `N`, in `main()`. The initial default value of `N` will be 0.
- **A function for each geometric shape**. Each function will take the previously entered integer value `N` and character value `C` as input parameters (Ensure that these values are valid before entering these functions). The return values of these functions will be `void`. The functions will print the respective geometric shape of `N` lines containing the input character `C`. `N` is considered the height of the shape. For a line, it is just printing the character in `C`. `N` number of times, so that it creates a vertically standing line of length `N`.

Example, for `N = 4` and `C = '*'`, the draw line function should output:

```
*
*
*
*
```

If a square is to be printed, then the following output is expected:

```
****
****
****
****
```

In case of a rectangle, we assume its width is `N+5`. It should look like the following:

```
*********
*********
*********
*********
```

If the user selects Triangle, then it should print a left justified triangle:

```
*
**
***
****
```

Suggested Steps to Complete the Assignment:

These steps will give you an idea on how to implement the solution for this assignment. Please note that this is a suggestion, you can follow another approach.

1. Create a directory named `lab3_<username>_<labsection>` and make it your current working directory.
2. Create a source file for this assignment named `lab3_<username>_<labsection>.c`
3. Write the function `menu()` and called it from `main()`, only prints the menu and by now the return value of `menu()` will be void. Compile and test it to ensure it works properly.
4. Change the return value of `menu()` to retrieve the character input within the function.
5. Add code to the `menu()` function to prompt and retrieve user input (allow any character), and return that input character to `main()`. Compile and test your code.
6. Enclose the print menu function and user input code within a loop that will exit the program when the Quit program choice is entered. Test the logic of your code to ensure that the loop only exits on proper input (`'q'` or `'Q'`).
7. Create six functions for the other (non-Quit) choices. Put a print statement such as "This function <*explanation*>" or some other informative statement in the body of the function. For functions that return a value, return any character or number. This will be changed when the function is filled in.
8. Within the loop in `main()`, create the logic to call each function based on input from the menu choice (and handle incorrect input choices). Test this logic by observing the output of the stub function statements.
9. Fill in the logic and code for each function. Note that the Line drawing function is probably a little easier (logically) to write than the Right Justified Printing function, so you may want to write it first. Once you have the Line drawing function complete, think about what additional character(s) you will have to print (and how many) to make a square shape. Step by step, you can write functions for other shapes as well. *This is the part of the assignment where you develop your skills to create algorithms that solve specific problems*.

10. Create a function that validates that `N` and `C` do not have their initial values. Call this function before drawing a shape and ask the user to fill in these values to draw a shape if at least one of them has its initial value.
11. Test your program.
12. Create a `Makefile` inside following the instructions given on "**Makefile**" section.
13. Perform instructions for submission.

Suggestions to Test your completed program:

A sample input file is included with the assignment. To test your program, you can use Unix redirection to enter the choices for your program automatically. To do this, type the name of your program, the "<" character (Unix **`stdin`** redirection), and the name of the sample input file:

```
$ lab3_<username>_<labsection> < lab3_input.txt
```

This will run your program with the input values provided in `lab3_input.txt`. You can also redirect output to a file with the ">" character (Unix **`stdout`** redirection):

```
$ lab3_<username>_<labsection> < lab3_input.txt > lab3_output.out
```

The above command will run your program using the sample input choices and save the resulting output in the file `lab3_output.out`. Make sure that your program runs correctly with the `lab3_input.txt` file.

Makefile:

Create a Makefile similar to the one you did for Lab 2

You will now create a target in your **`Makefile`** using a variable.
Add the following line below the `CFLAGS` line you added previously:

```
TARGET = lab3_<username>_<labsection>
```

Modify your `all:` target as follows:

```
all: $(TARGET).c
```

Modify it again as follows:

```
$(TARGET): $(TARGET).c
```

Next, edit the old compile line so that it references the new variable names you created:

```
$(CC) $(TARGET).c -o $(TARGET) $(CFLAGS)
```

Don't forget the **`tab`** before the `$(CC)`

Finally, edit the `clean:` target so that it removes the executable by its variable name rather than the explicit name:

```
rm $(TARGET)
```

Don't forget the **tab** before the `rm`

Once that is done, you can compile your executable by running the **`make`** command. Test this to ensure that your **`Makefile`** works correctly.

Submission:

You will submit a typescript file containing a listing of your code (showing that it compiles without errors) and a run of the program after compiling. Follow this procedure:

1. Create a typescript file named `lab3_typescript_<username>_<labsection>`
2. Show that you are logged onto Zeus. (`uname -a`)
3. Show a listing of your directory
4. Show a listing of your code
5. Remove any versions of the executable that may appear. Use your **`Makefile`** for this.
6. Compile the code using your **`Makefile`**
7. Show that the executable file was created from the compile command
8. Run the code using redirection "<" and use the `lab3_input.txt` file as input
9. Remove the executable using your **`Makefile`**
10. End the typescript
11. Be sure your directory ONLY contains the source file, script and Makefile
12. Verify your typescript file is correct, then change (**`cd`**) to the directory above
13. Create a **`tarfile`** of your `lab3_<username>_<labsection>` directory
14. Submit the **`tarfile`** to Blackboard

Points to Review:

- **Learn a bit more about Unix Redirection**: A straightforward page that discusses I/O redirection can be found at: http://www.ee.surrey.ac.uk/Teaching/Unix/unix3.html
- Note the differences between the `while` loop and the `do-while` loop.
  Which one is guaranteed to execute the code within the loop at least once?
- Be aware that the `switch` statement needs `break` statements to avoid "falling through" to the next case.

Congratulations! You have completed your assignment 😎

Errata:

[Feb, 14] Page 2/4

- The figure of the Triangle with **n = 4** was updated.