CISC105 Spring 2007 Lab04

- Review the code examples from class.
- Familiarize yourself with the website's section on Academic Honesty. I require strict adherence to this policy, and I will pursue any violations through the University's process outlined in the student manual online. If you are freaking out about getting a lab done, come see your prof or TA, even if it means being a day or two late - the penalties for lateness are much nicer than the penalties for sharing code or otherwise cheating.
- Some programs below are associated with a question. **Answer the questions** using C comments below your code in the program file.
- Any time you expect user input, you must print a **prompt** telling the user what to enter.

Programs

For each numbered problem below you will write a small program. Name each program lab04.n.c, where n is the number in the list below. For example, the name of the file for the first will be lab04.1.c

1. Copy the program from lab three that had a function that printed a line of asterisks. Modify the function to make a square of asterisks as follows. This time, instead of looping up to the user input number, loop until the input number squared (i.e. print enough asterisks to fill a square of size "input"). Once that is working, put a statement inside the loop that uses a conditional to print a single newline. The condition should be true when the loop reaches the end of one line of a square. For example, if the user enters "6", then the program will print 36 separate asterisks with a newline after every sixth asterisk.

```
> ./a.out
How big would you like your square? 3
***
***
***
How big would you like your square? -1
Goodbye!
>
```

2. Here is a different way to make a square of asterisks. Write another function that will get called from main() with the size of a square. **If** the size of the square is to be four, it will call the asterisk line function from problem 1 four times.

```
> ./a.out
What size would you like your square? 4
****
****
****
****
What size would you like your square? -1
Goodbye!
>
```

3. Copy the program you made for problem 2. Using the same function you wrote for that problem, you will now make a triangle of asterisks. **If** the size of the square is to be four, it will call the asterisk line function four times.

```
> ./a.out
What size would you like your triangle? 3
*
**
***
What size would you like your triangle? 4
*
**
***
****
What size would you like your triangle? -1
Goodbye!
>
```

4. Place the functions you have written for 2 and 3 into a new program, and give the user a menu that allows the user to select an action or exit. Use a switch statement in main() to implement the menu. The sentinel loop should run until the user enters -1, but replace the -1 in your program with a defined constant called STOP by adding a preprocessor directive (see section 2.1 of H&K).

```
> ./a.out
Type -1 to exit, 1 to print a square, or 2 to print a triangle: 2
What size would you like your triangle? 3
*
**
***
Type -1 to exit, 1 to print a square, or 2 to print a triangle: 1
What size would you like your square? 2
**
**
Type -1 to exit, 1 to print a square, or 2 to print a triangle: -1
Goodbye!
>
```

5. Write a function called **indent**. It will be a void function and takes a single integer parameter called **length**. If the number is negative, the function prints "error" and returns. Otherwise, it prints **length** spaces (nothing else) and then returns.

6. Write a program that uses **indent** to create the indents for a diagonal line. Your program will use a for loop that calls indent to print the following (based on user input):

```
% ./a.out
Enter height: 5
*
 *
  *
   *
    *
%
```

7. Declare an integer variable **sum**. Using a while loop, take a series of five integers from the user, and add each one to sum. After the loop is over, print the sum:

```
> a.out
Please enter an integer: 2
Please enter an integer: 4
Please enter an integer: 3
Please enter an integer: 1
Please enter an integer: 2
The sum is 12
>
```

8. Write a program that takes positive integers from the user until the user enters a sentinel -1, and then prints the average of the integers entered. Use a unary type cast so that the correct kind of division is performed. The sentinel should not be considered data.

9. Declare an array of ten integers, and use a for loop to initialize the array to the integers 10-19. Print the array using a different for loop.

10. Declare an array of ten integers, and use a for loop to initialize the array ten numbers entered by the user. Print the array using a different for loop.

You should have a total of 10 programs named lab04.1.c to lab04.10.c. Make a single script file (see lab00 for the scripting instructions) where you cat, compile, and run each one in its final form (if it didn't compile, don't run it in the script - mark the place in the printed script file with a colored marker so it stands out). After all files have been run in the script, use ls and cd to show your new directories and their files in the script.

On the first page of every printed copy for this course, your name, section, and TA's name must appear.

Submit all program and script files on MyCourses before midnight Thursday of next week, and give the paper version to your TA at the beginning of your Friday lab (or in lecture Friday if you have a Wednesday lab). Note: cat, compile, and run each program in order! Do *not* cat all programs, then compile, etc.