APS 105 — Computer Fundamentals
Lab 5: Functions, Logic, and Debugging
Winter 2018

The goal of this laboratory is to practice the material on functions. You are to write one C program that consists of a ***main()*** and several functions. The program displays the Pascal's triangle as explained in the following section.

Instructions:

- Similar to previous labs, your solution will be marked by your TA during your scheduled lab period for programming style and your understanding, **and** should also be submitted electronically by the end of your scheduled lab period.

Preparation

Read through this entire document carefully, and do the work to create the programs that are described below. You are encouraged to:

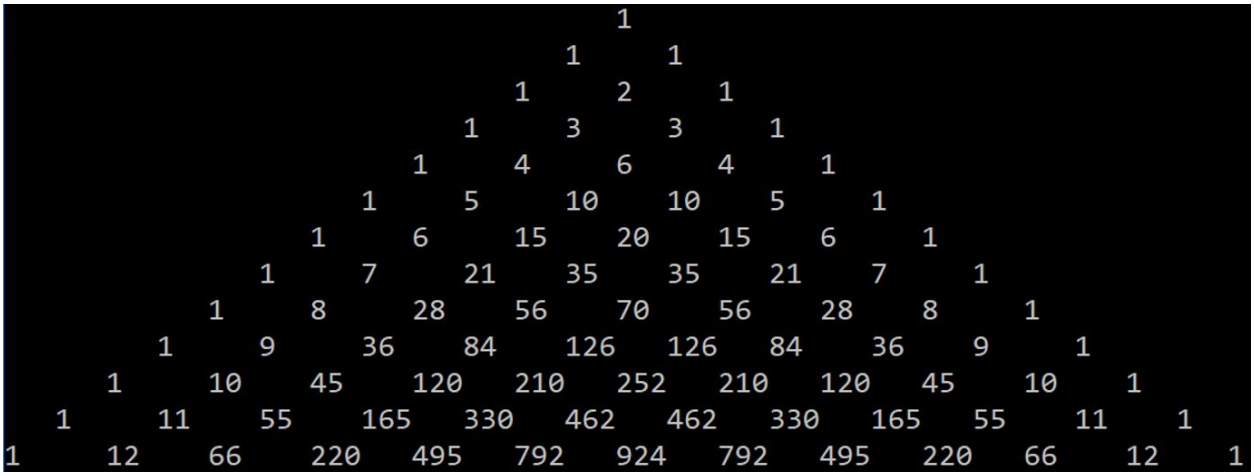
- Read the class bulletin board on Piazza, to see if others had similar problems. If you do not see anything helpful there, ask a question. Do not ask for, or ever give a full or partial solution to the lab assignment.
- Ask for assistance from your lab/tutorial TAs.
- Attend and ask questions during the plenary sessions.

Notes:

- In the sample output examples that follow:
  - The text <**enter**> stands for the user pressing the enter key on the keyboard. On some keyboards, the enter key may be labeled return.
- Throughout this lab, there is a single space after the colon (:) in each output line.

Pascal's Triangle

In a file called **Lab5.c**, write a C function named "***triangle***" that outputs Pascal's triangle (for example) as follows:



In general the Pascal's triangle can be represented as:

```
                0C0
              1C1 1C0
            2C2 2C1 2C0
          3C3 3C2 3C1 3C0
        4C4 4C3 4C2 4C1 4C0
                ...
```

where *n**C**r* represents how many ways there are to choose ***r*** from ***n***, not counting duplicates.
In mathematics, it is usually presented as $\binom{n}{r}$. The formula used to calculate *n**C**r* can be written as:

$$n\boldsymbol{C}r = \frac{n!}{r!\,(n-r)!}$$

where $n!$ is the factorial of n.

The function ***triangle***:

- Is called *by value* where exactly one parameter (the number of rows of the Pascal's triangle) is passed to it.
- Returns void (i.e. no value).
- Prints out the Pascal's triangle to the standard output.
- Employs two other functions namely:

```
int choose(int n, int r);
```

  - That chooses r from n

```
int factorial(int n);
```

  - That calculates factorial of n.

Note: You are allowed to use any other function as need be.

You are required to provide a complete C program by writing the code for the ***main()*** function

- Prompts user to supply the number of rows in the Pascal's triangle.
- Calls the function ***triangle*** such to displays the Pascal's triangle based on the number of rows provided by the user.
- Terminates the run of the program is the user supplies a negative value.

An example of the required output from your program is as follows:

```
Enter the number of rows: 0
Enter the number of rows: 1
1
Enter the number of rows: 2
   1
1     1
```

```
Enter the number of rows: 3
      1
   1     1
1     2     1
Enter the number of rows: 4
         1
      1     1
   1     2     1
1     3     3     1
Enter the number of rows: 5
            1
         1     1
      1     2     1
   1     3     3     1
1     4     6     4     1
```

```
Enter the number of rows: 6
               1
            1     1
         1     2     1
      1     3     3     1
   1     4     6     4     1
1     5     10    10    5     1
Enter the number of rows: 7
                  1
               1     1
            1     2     1
         1     3     3     1
      1     4     6     4     1
   1     5     10    10    5     1
1     6     15    20    15    6     1
```

```
Enter the number of rows: 8
                     1
                  1     1
               1     2     1
            1     3     3     1
         1     4     6     4     1
      1     5     10    10    5     1
   1     6     15    20    15    6     1
1     7     21    35    35    21    7     1
```

Notes:

- The number of the rows is limited to integers between ***0*** and ***13*** inclusive.
- Automarker tests inclusively between ***0*** and ***13***.
- If ***0*** then there are no triangles displayed.
- If negative integer then program terminates.
- The number displayed in the last column MUST be always displayed at the start of the line. In other words and as an example:
    - If the number of rows entered is ***3*** then the first row displays its number at column ***7***.
    - Carefully calculate the spaces used between the numbers so that the numbers are properly aligned across rows.
    - Use of a function ***spaces()*** that deals with spacing is recommended.

Grading by TA and submitting your program for Auto-Marking

There are a total of 10 marks available in this lab, marked in two different ways:

1. **By your TA, for 4 marks out of 10.** Once you are ready, show your program to your TA so that we can mark your program for style, and to ask you a few questions to test your understanding of what is happening. Programs with good style have been described in class, but briefly, they are:

    a. Clear comments that describe what is happening in the program.

    b. Good choices for variable names that indicate their purpose. Please adopt the following naming convention illustrated above — if have you a variable that is described by multiple words, user lower case for the first letter of the first word, and Upper case for all subsequent words, e.g. **blendMode**.

    c. Properly indented code that has appropriate spacing between lines for readability.

    The TA will also ask you some questions to be sure that you understand the underlying concepts being exercised in this lab.

2. **By an auto-marking program, for 6 marks out of 10.** You should run the following command:

    /share/copy/aps105s/lab5/exercise

    Within the directory that contains your solution programs.

    This program will look for the files **Lab5.c** your directory, compile them, and run them on some test cases. If there is anything wrong, the **exercise** program will report this to you, so read its output carefully, and fix the errors that it reports.

    A key part of what you are to learn in this lab is the use of this program - most software programs give this kind of report. Read through the output of the **exercise** program and see if it is happy with everything, or if it is reporting an error. If there is an error, it will say so, and then it is up to you to fix what is wrong and try again. You'll need to do this for every subsequent lab.

    Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period as that is

the due time. To do so, go into the directory containing your **Lab5.c** file (and make sure this is the right place!) and type the following command:

/share/copy/aps105s/lab5/submit

This command will re-run the exercise program to check that everything looks OK. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

Important Note: You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

/share/copy/aps105s/lab5/viewsubmitted

After the Final Deadline — Obtaining Automark

After all lab sections have finished, a short time later, you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so run the following command:

/share/copy/aps105s/lab5/marker

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases' output and what went right or wrong.

Good Luck!