NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
ISLAMABAD CAMPUS
PROGRAMMING FUNDAMENTALS (CS118) – FALL 2018
ASSIGNMENT-4

Due Date: October 27, 2018 (11:30pm)

Instructions:

1. *Write the C++ programs for all the question.*
2. *Solution to all the problems should be written in a separate (.cpp) file.*
3. *Submit the source code via **Google classroom**. Submissions via email will not be accepted.*
4. *Use proper naming convention to name the file containing source code.*
   For example, the file containing the source code for first question of the first assignment should be named as i18xxxx_assignment4_q1.pp, replace i18xxxx with your roll number.
5. *Place all your files in a folder. Zip it, rename the folder with your roll number and upload that folder on Google classroom*
6. ***The output should be well presented.** There will be marks of the presentation.*
7. ***Use proper checks where required**. There will be marks of proper checks too.*
8. *Do not **plagiarize**. Use efficient, simple and clean logics and codes.*
9. *Use proper **indentation** in your code. Indentation improves **readability** and helps in **debugging**.*
10. *Use appropriate naming conventions for **variable names**.*
11. *Note: You have to follow the submission instructions to the letter. Failing to do so can get a zero in assignment. We are not going accept any file without the specified naming convention whatever the reason will be.*

1. Write while loop to find the minimum point of following function. *[Hint: Evaluate the function by writing nested while loops for a large domain of x and y values and find the values where minimum value occur]*.

$$f(x,y)=(1-x)^2+100(y-x^2)^2$$

2. Using for loop write a C++ program get mean and average of the scores of student.

3. One interesting application of computers is drawing graphs and bar charts (sometimes called “histograms”). Write a program that reads five numbers (each between 1 and 30). For each number read, your program should print a line containing that number of adjacent asterisks. For example, if your program reads the number seven, it should print *******.

4. (Calculating the Value of π) Calculate the value of π from the infinite series Print a table that shows the value of π approximated by one term of this series, by two terms, by three terms, and so on. How many terms of this series do you have to use before you first get 3.14?

$$\Pi = 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}\ \ldots\ldots$$

5. Implement the ***Euclidean Algorithm*** for finding the greatest common divisor of two given positive integers. This algorithm transforms a pair of positive integers (*m*, *n*) into a pair (*d*, 0) by repeatedly dividing the larger integer by the smaller integer and replacing the larger with the remainder. When the remainder is 0, the other integer in the pair will be the greatest common divisor of the original pair (and of all the intermediate pairs). For example, if *m* is 532 and *n* is 112, then the Euclidean Algorithm reduces the pair (532,112) to (28,0) by (532,112) → (112,84) → (84,28)→ (28,0). So 28 is the greatest common divisor of 532 and 112.

6. Write for loop which asks user to enter an integer and reveres that integer number. Your program also calculates sum of digits that integer number. For example if User enters 1234 your program will print 4321 and sum = 10. Remember this should be done using while loop.

7. Write a while loop that allows a user to convert positive numbers from binary (base 2) to decimal (Base 10). Use simple and accurate algorithm to produce the correct result. Here is an example of an algorithm illustrating how to convert from binary (base 2) to

Decimal:

(Base 10):

The binary or base-2 number 1 0 0 1 0 1 0 1 can be converted to a decimal number as follows:

$$1*2^7+0*2^6+0*2^5+1*2^4+0*2^3+1*2^2+0*2^1+1*2^0$$

= 128 + 16 + 4 + 1

= 149

The binary number 1 0 0 1 0 1 0 1 is written as 149 in the decimal system.

8. Write for loop that accepts two integer input “lines” and “cheers” and prints a series of "cheer" lines at increasing levels of indentation. The first parameter represents the number of lines of output to print, and the second represents the number of "cheers" per line. For example, if lines=2 and cheers =4 than you should print 2 lines of output, each containing 4 "cheers." A "cheer" is an occurrence of the word "Go" in the output.

Neighboring cheers are separated by the word "Buddy" (of course you can put name of your favorite political person here as well :)), so 1 cheer is printed as "Go", 2 cheers as "Go Buddy Go", 3 cheers are printed as "Go Buddy Go Buddy Go", and so on.

The lines you print should be displayed at increasing levels of indentation. The first line displayed should have no indentation, but each following line should be intended by 3 spaces more than the one before it. In other words, the 2nd line of output should be indented by 3 spaces, the 3rd line by 6 spaces, and so on. You may assume that both parameters passed your function will have values of at least

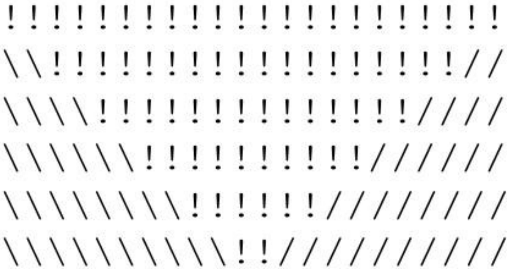
INPUT:

```
Lines=2                 Lines=4                  Lines=2
Cheers=1                Cheers=3                 Cheers=4
```

OUTPUT

```
Go       Go Buddy Go Buddy Go          Go Buddy Go Buddy Go Buddy Go
   Go       Go Buddy Go Buddy Go          Go Buddy Go Buddy Go Buddy Go
               Go Buddy Go Buddy Go
                  Go Buddy Go Buddy Go
```

9. Write for loop that produces the following output. Use nested while loops to capture the structure of the figure.

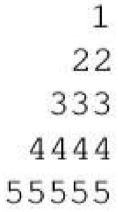


```
!!!!!!!!!!!!!!!!!!!!!!
\\!!!!!!!!!!!!!!!!!!//
\\\\!!!!!!!!!!!!!!////
\\\\\\!!!!!!!!!!//////
\\\\\\\\!!!!!!////////
\\\\\\\\\\!!//////////
```

10. Write while loops to produce the following output:

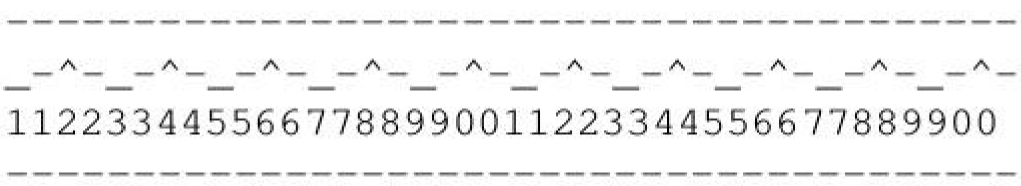
```
##
#  #
#     #
#        #
#           #
#               #
```

11. Write while loops to produce the following output:



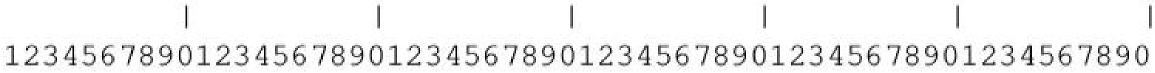
```
    1
   22
  333
 4444
55555
```

12. Write for loops to produce the following output, with each line 40 characters wide:



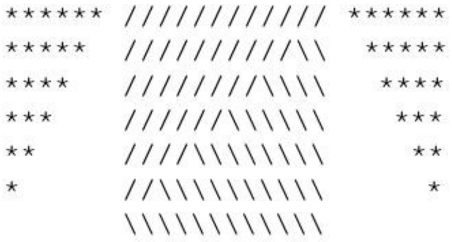
```
----------------------------------------
_-^-_-^-_-^-_-^-_-^-_-^-_-^-_-^-_-^-_-^-
1122334455667788990011223344556677889900
----------------------------------------
```

13. It's common to print a rotating, increasing list of single-digit numbers at the start of a program's output as a visual guide to number the columns of the output to follow. With this in mind, write nested while loops to produce the following output, with each line 60 characters wide:



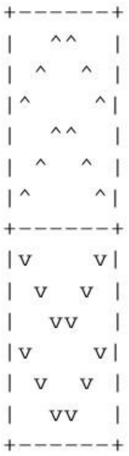
```
         |         |         |         |         |         |
123456789012345678901234567890123456789012345678901234567890
```

14. Write a for loop that produces the following output (with loops):



```
****** //////////// ******
*****  //////////\\  *****
****   ////////\\\\   ****
***    //////\\\\\\    ***
**     ////\\\\\\\\     **
*      //\\\\\\\\\\      *
       \\\\\\\\\\\\
```

15. Write a program that produces the following output (with loops, **Remember Divide and Conquer**):



```
+------+
|  ^^  |
| ^  ^ |
|^    ^|
|  ^^  |
| ^  ^ |
|^    ^|
+------+
|v    v|
| v  v |
|  vv  |
|v    v|
| v  v |
|  vv  |
+------+
```

16. PakAsia is a country that has currency notes of 6, 9 and 20 PakAsia Rupees (PAR) only. Thus, it is possible, for example, to exchange exactly 15 Pakistani Rupees (PKR) (with one note of 6 Rs and a second note of 9 Rs will add up to make 15, since 1PKR == 1PAR), but it is not possible to exchange exactly 16 PKR, since no non-negative integer combination of 6's, 9's and 20's notes of PAR add up to make 16 PKR. To determine if it is possible to exchange exactly **n** PKR to PAR, one has to find non-negative integer (can be 0) values of a, b, and c such that:

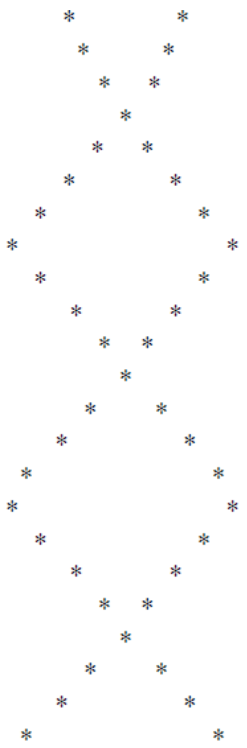
$$\mathbf{6a+9b+20c=n}$$

Now write a code that takes, 'n' PKR from user as input, and prints if it is possible to exchange it with a combination of 6, 9 and 20 PAR such that the total sum of PakAsia Rupees equals **n**, and otherwise prints it is not possible.

17. Write a while loop that produces the following output:

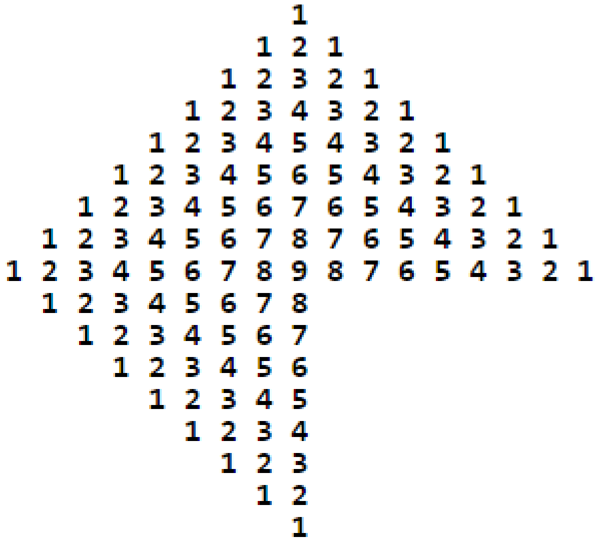
$$\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \frac{15}{16}, \frac{31}{32} \dots \dots \dots.$$

18. Write a program that produces the following output (with loops):



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

19. Write a program that produces the following output (with loops):



```
                1
              1 2 1
            1 2 3 2 1
          1 2 3 4 3 2 1
        1 2 3 4 5 4 3 2 1
      1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8
    1 2 3 4 5 6 7
      1 2 3 4 5 6
        1 2 3 4 5
          1 2 3 4
            1 2 3
              1 2
                1
```