PF Fall 2021 - Assignment 3

Submission Instructions:

1) You can email me (aamina.batool@lhr.nu.edu.pk) or your TA Salman (l181127@lhr.nu.edu.pk), if you have any query.
2) Plagiarism tool will be used to check if you have cheated off of internet or any other source. If someone's assignment is plagiarized, that student will be given F straight away so beware.
3) Make separate .cpp files for every question. Kindly name your .cpp files as Q1_21L-XXXX. (XXXX here would be replaced by your roll number).
4) Add cpp files of all the questions in a folder. Name your folder by including your roll number as 21L-XXXX-A2 (A2 here means assignment 2) and then compress the folder as .rar, and submit it.
5) And kindly note that write your roll no. in the format given above and not as L21-XXXX or 21-XXXX because it makes it hard to find the submission of specific student.
6) There is straight **20 Percent deduction** of marks for students who submit more than 1 folder.
7) If any student resubmits make sure they remove the folder they last submitted from drive, by clicking the cross that shows when you have attached a file at Submission folder.
8) You have to upload your assignments in google classroom. Don't forget to turn in after uploading your assignment.
9) The deadline for this assignment is **November 27, 2021, 11:59 pm**. No late submission will be accepted.

Reading Assignment: Study, solve and understand example 6-5 and 6-5 of chapter 6.

Question 1.

Write a function **isPrime** to find whether a given integer is prime or not. If number is prime, your function should return true and false otherwise. Write a main function and call your function 5 times, inside main function, for 5 different integers. After every call, print the result i.e. number is prime or not prime.

Question 2.

The game of "**23**" is a two-player game that begins with a pile of 23 toothpicks. Players take turns, withdrawing either 1, 2, or 3 toothpicks at a time. The player to withdraw the last toothpick loses the game. Write a C++ function **twothree()** for human vs. computer program that plays "23". The human should always move first.

Write a function **human_Turn()** which will take input from player. When the human player enters the number of toothpicks to withdraw, the program should perform input validation. Make sure that the entered number is between 1 and 3 and that the player is not trying to withdraw more toothpicks than exist in the pile.
Write a function **computer_Turn()**, when it is the computer's turn, it should play according to the following rules:

a. If there are more than 4 toothpicks left, then the computer should withdraw 4 – X toothpicks, where X is the number of toothpicks the human withdrew on the previous turn.

b. If there are 2 to 4 toothpicks left, then the computer should withdraw enough toothpicks to leave 1.

c. If there is 1 toothpick left, then the computer has to take it and loses.

Note: function **twothree()** must use helper functions human_Turn and computer_Turn, for the game.

Question 3:

It is known from Babylonian times that the square root of a number can be approximated by a method now known as 'divide and average' method. Since the Babylonians used it, it is also called the Babylonian algorithm.
METHOD: Suppose we want to compute the square root of a number N.
Let A>0 be a guess for square root (N), then a better approximation is given by: B=(A+N/A)/2.
We can then improve the approximation B using C=(B+N/B)/2.
EXAMPLE: For example, lets compute: square root of N=2.
Let our initial guess be 1.
The next better approximation is (1+2/1)/2=1.5
The next better approximation is (1.5+2/1.5)/2=1.416667
The next better approximation is (1.416667+2/1.416667)/2=1.414216
And so on. The more you repeat this, the closer you will get to the actual answer. Just keep in mind that the square root of 2 is an irrational number, which means that you can keep on improving your approximation forever.
1. Write a C++ Function square_root (int n) to implement above mentioned method of calculating square root and will return the calculated square root. You have to run 10 iterations for approximation.

Question 4: (Factorial)

1. Write a function that returns the factorial of a number.

Note: The factorial of a number is the product of all the integers from 1 to that number. For example, the factorial of 6, 6! = 1*2*3*4*5*6 = 720. Factorial is not defined for negative numbers, and the factorial of zero is one, 0! = 1
2. Write a function which will take two inputs and find **nPr** (number of permutations).

nPr (n, r) = n! / (n-r)!

Use Factorial function designed in part 1

3. Write a function which will take two inputs and find **nCr** (number of combinations)

$$nCr(n, r) = n! / (n-r)! * r!$$

Use Factorial function designed in part 1

4. Write a function which will take two inputs and find **nCr** (number of combinations) where

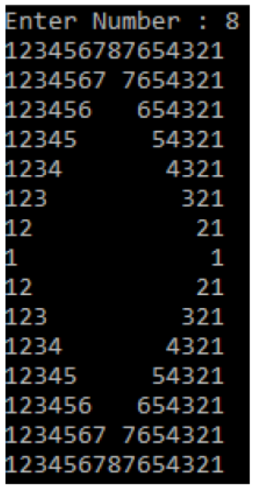
$$nCr(n,r) = nPr(n,r) / r!$$

Use Factorial function designed in part 1 and nPr of part 2

5. Design a menu function that will take a number from the user as input and then perform specific operation based on input value. All cin and cout should only be done in the menu function. a. On ‘1’ print factorial of number.
   b. On ‘2’ print nPr.
   c. On ‘3’ print nCr designed in part 3.
   d. On ‘4’ print nCr designed in part 4.
   e. On ‘5’ Exit Program.

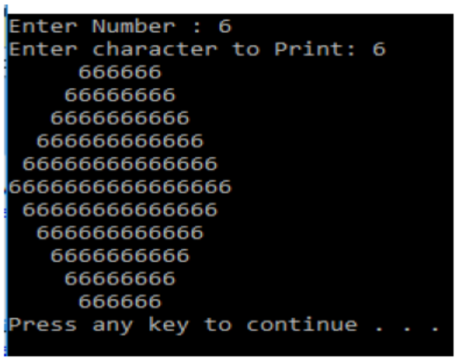
Question 5:

Draw following Patterns for positive values of integers provided by user as input.



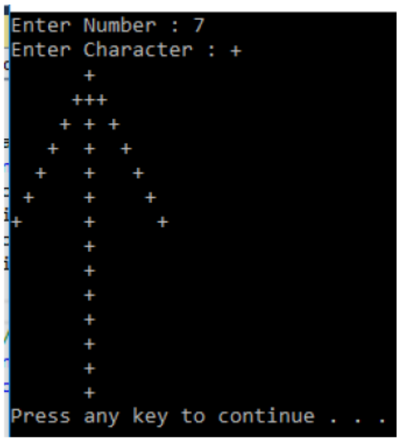
```
Enter Number : 8
123456787654321
1234567 7654321
123456   654321
12345     54321
1234       4321
123         321
12           21
1             1
12           21
123         321
1234       4321
12345     54321
123456   654321
1234567 7654321
123456787654321
```

A holo square of numbers with a diamond.



```
Enter Number : 6
Enter character to Print: 6
     666666
    66666666
   6666666666
  666666666666
 66666666666666
6666666666666666
 66666666666666
  666666666666
   6666666666
    66666666
     666666
Press any key to continue . . .
```

A Hexagone of any size and chracters provided by user.



```
Enter Number : 7
Enter Character : +
      +
     +++
    + + +
   +  +  +
  +   +   +
 +    +    +
+     +     +
      +
      +
      +
      +
      +
      +
      +
Press any key to continue . . .
```

An arrow sign of provided size and character.