**National University of Computer and Emerging Sciences, Lahore Campus**

**Course: Programming Fundamentals Course Code:**

**CS 2000**

**Program: BS (CS) Semester: Fall 2023 Due Date October 20,2023 Total Marks: 250** 

**Section: B Page(s): 6**

**Type: Assignment 3**

**Question # 1:**

Write a program to simulate a very simple version of Snakes & Ladders. This game contains no snakes and no ladders. Two players compete in this game and the first player that reaches 20 points wins the game. The gameplay is really simple: two players keep on rolling the dice in

turn until one of them reaches exact 20 points. If a player rolls a dice and the total points cross 20, this turn is discarded, and the total points don’t change.

User is prompted to press any key to roll the dice (explore *getch()* to implement this behavior). You can use *rand()* to implement rolling a dice functionality.

A sample run of this program is given below:

| Player 1’s turn: 5  Player 1’s total: 5  Player 2’s turn: 4  Player 2’s total: 4  Player 1’s turn: 2  Player 1’s total: 7  Player 2’s turn: 1  Player 2’s total: 5  Player 1’s turn: 5  Player 1’s total: 12  Player 2’s turn: 6  Player 2’s total: 11  Player 1’s turn: 1  Player 1’s total: 13  Player 2’s turn: 3  Player 2’s total: 14  Player 1’s turn: 4  Player 1’s total: 17  Player 2’s turn: 5  Player 2’s total: 19 |
| --- |

| Player 1’s turn: 1  Player 1’s total: 18  Player 2’s turn: 3  Player 2’s total: 19 (move not possible)  Player 1’s turn: 2  Player 1’s total: 20  Player 1 Won!!! |
| --- |

**Question 2**

Write a program in C to Check Whether a Number can be Express as Sum of Two Prime Numbers.

***Test Data :***

Input a positive integer: 16

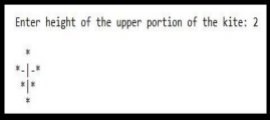
*Expected Output* :

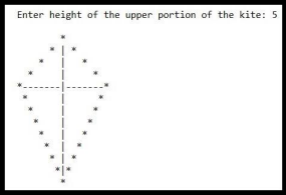
16 = 3 + 13

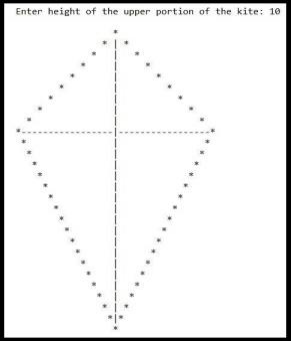
16 = 5 + 11

**Question 3**

Write a C++ **code** to print the pattern of a kite. The program takes the height of the upper portion of the kite as parameter. Some samples are given below:







**Question 4**

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, let’s 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++ code that takes a number N and implements above mentioned method of calculating the square root and prints the calculated final square root. You have to run 10 iterations for approximation.

**2.** Keep repeating the code until the user enters ‘-1’ for terminating the program **Question 5**

Write a C++ program for the following:

**1.** 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 code which will take two inputs and find nPr (number of permutations). **nPr (n, r) = n! / (n-r)!**

**3.** Write code which will take two inputs and find nCr (number of combinations) **nCr (n, r) = n! / (n-r)! \* r!**

**4.** Write a function which will take two inputs and find nCr (number of combinations) where **nCr(n,r) = nPr(n,r) / r!**

**5.** Design a menu function that will take a number from the user as input and then perform specific operation based on input value.

**Sample Output:**

**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 6**

The last digit of a credit card number is the *check digit*, which protects against transcription errors such as an error in a single digit or switching two digits. The following method is used to verify actual credit card numbers but, for simplicity, we will describe it for numbers with 8 digits instead of 16:

∙ Starting from the rightmost digit, form the sum of every other digit. For example, if the credit card number is 4358 9795, then you form the sum 5 + 7 + 8 + 3 = 23.

∙ Double each of the digits that were not included in the preceding step. Add all digits of the resulting numbers. For example, with the number given above, doubling the digits, starting with the next-to-last one, yields 18 18 10 8. Adding all digits in these values

yields 1 + 8 + 1 + 8 + 1 + 0 + 8 = 27.

∙ Add the sums of the two preceding steps. If the last digit of the result is 0, the number is valid. In our case, 23 + 27 = 50, so the number is valid.

Write a program that implements this algorithm. The user should supply an 8-digit number, and you should print out whether the number is valid or not. If it is not valid, you should print the value of the check digit that would make it valid.

**Question 7**

The following experiment was devised by Comte Georges-Louis Leclerc de Buffon (1707– 1788), a French naturalist. A needle of length 1 inch is dropped onto paper that is ruled with lines 2 inches apart. If the needle drops onto a line, we count it as a *hit*. (See 1 & 2). Buffon discovered that the quotient tries/hits approximates p.

For the Buffon needle experiment, you must generate two random numbers: one to describe the starting position and one to describe the angle of the needle with the *x*-axis. Then you need to test whether the needle touches a grid line.

Generate the *lower* point of the needle. Its *x*-coordinate is irrelevant, and you may assume its y coordinate to be any random number between 0 and 2. The angle α between the needle and the *x* axis can be any value between 0 degrees and 180 degrees (π radians). The upper end of the needle has *y*-coordinate



The needle is a hit if yhigh is at least 2, as shown in Figure 9. Stop after 500 tries and print the quotient *tries*/*hits*. (This program is not suitable for computing the value of π. You need π in the computation of the angle.)

| **Figure 1** The Buffon Needle Experiment  **Figure 2** A Hit in the Buffon Needle Experiment |
| --- |

**Question 8**

This is a well-known game with a number of variants. The following variant has an interesting winning strategy. Two players alternately take marbles from a pile. In each move, a player chooses how many marbles to take. The player must take at least one but at most half of the marbles. Then the other player takes a turn. The player who takes the last marble loses.

Write a program in which the computer plays against a human opponent. Generate a random integer between 10 and 100 to denote the initial size of the pile. Generate a random integer between 0 and 1 to decide whether the computer or the human takes the first turn. Generate a random integer between 0 and 1 to decide whether the computer plays ***smart*** or ***stupid*.** In stupid mode the computer simply takes a random legal value (between 1 and *n* / 2) from the pile whenever it has a turn. In smart mode the computer takes off enough marbles to make the size of the pile a power of two minus 1—that is, 3, 7, 15, 31, or 63. That is always a legal move, except when the size of the pile is currently one less than a power of two. In that case, the computer makes a random legal move.

You will note that the computer cannot be beaten in smart mode when it has the first move, unless the pile size happens to be 15, 31, or 63. Of course, a human player who has the first turn and knows the winning strategy can win against the computer.

**Question 9**

The following iterative sequence is defined for the set of positive integers:



Using the rule above and starting with 13, we generate the following sequence: 

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1. Which starting number**,** ≤ N produces the longest chain? If many possible such numbers are there print the maximum one.

In simple words, input N from the user and tell in the range from 1 to N at which **n** it has the largest sequence.

**Note**: Once the chain starts the terms are allowed to go above.

**Example**:

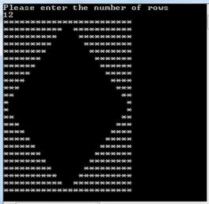
**Input**=10

**Output**=9

**Explanation**:

**Question 10**

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

****

**Part B**

For rows=10

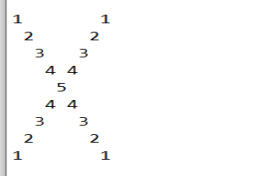


**Part C**

****

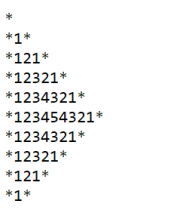
**Part D:**

**For N=5**

****

**Part E:**

**For N=5**

****

Departmentof Computer Science, FASTSchoolof Computing FAST-NU, Lahore Page6