National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Programming Fundamentals**	**Course Code:**	**CS 1002**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2022**
Due Date	**30 Oct 2022**	**Total Marks:**	**50**
Section:	**CS-1J & SE-1C**	**Page(s):**	**2**
Exam:	**Assignment#1**		

Task#1: 08 Marks

Write a program that prompts the user to enter starting and ending range of the series, where both the values should be positive and ending range should be greater than the starting range. Your task is to count and print all the palindromes and prime numbers in the given range using nested loops. Input validations is mandatory.

Task#2: 08 Marks

We know that according to Pythagoras theorem (base^2) + (perpendicular^2) is equals to (hypotenous^2). Write a program that prompt the user to enter starting and ending range where ending range (input validation criteria is similar to task1). Your task is to determine all such combinations satisfying the Pythagoras theorem using nested loops e.g.,3^2+4^2 = 5^2 and 6^2 + 8^2 = 10^2, so (3,4,5 and 6,8,10) are the two combinations in the range 3 to 10.

Task#3: 04 Marks

Write a program that generates a random number and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." The program should use a loop that repeats until the user correctly guesses the random number. Your program should also keep a count of the number of guesses that the user makes. When the user correctly guesses the random number, the program should display the number of guesses.

Task#4: 30 Marks

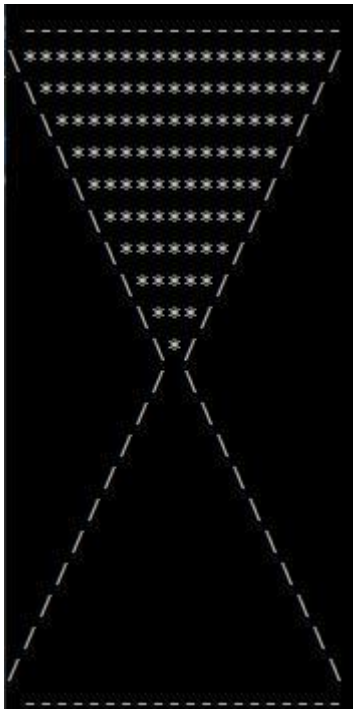
Write a program to simulate hourglass. Get the following input from user.

`Time:` Total time (in seconds) that can be measured/simulated using this hourglass. (Remember: Time can't be 0 or negative)

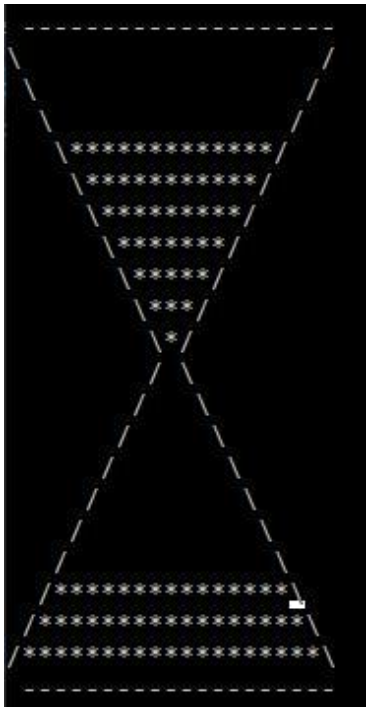
`C:` The character to represent sand in the hourglass.

The simulation works in this way: after one second, some sand from the upper portion of the clock drops into the lower portion of the clock. For your program, you can implement it in this way: after every second, remove one row from the upper portion of the hourglass and add a new row in the lower portion of the hourglass. The simulation stops when no sand remains in the upper portion of the hourglass. You can use the **Sleep** function to stop execution of the program for 1 second.

Some illustrations of hourglass are given on the next page:



totalTime: 10, C: ‘*’



After 3 seconds

totalTime: 5, C: ‘#’

After 5 seconds