SSN COLLEGE OF ENGINEERING (Autonomous)
Affiliated to Anna University, Chennai
DEPARTMENT OF CSE

UCS 1211 PROGRAMMING IN C LABORATORY
A2 : Modular Programming with C

==========

Learning Outcome :

To be proficient in using functions in C
- input parameters (Call by value)
- input-output parameters (Call by reference)
- recursive functions

To declare the functions explicitly

To minimize the usage of global variables

To learn to develop code incrementally

Write the algorithm to solve the following problems and implement them in C. Solving any 5 of the following problems is mandatory.

1. Modify A1(1) to have a function *CheckOddEven(num)* that checks if the *num* is odd or even; sets a flag accordingly and return it. Use this function to find the sum of even and odd numbers in a given input of N numbers.
2. Write a C function *ReverseNum(num)* that takes integer *num* and reverses its digits. Let *num* be passed by reference.
   Example:
   Input: 453275
   Output: 572354
3. Write a function *power(X,N)* that will allow a floating-point number to be raised to an integer power. $Y = X^N$
   In other words, evaluate the formula where y and x are floating-point variables and n is an integer variable. Write a C program that will read in numerical values for x and n, evaluate the formula using *power(X,N),* and then display the calculated result. Test the program using the following data:

x	n	x	n
2	3	1.5	3
2	12	1.5	10
2	-5	1.5	-5
-3	3	0.2	3
-3	7	0.2	5
-3	-5	0.2	-5

4. Find the product of n floating point numbers. The numbers should be read from the keyboard. You should not use any looping construct. [Hint: use recursion and decide a suitable sentinel for termination of recursion.]
5. Write a recursive function that reads N and prints from N to 0.

Input	Output
10	9876543210

6. **Factorials**

The factorial of an integer n, written n!, is the product of all the integers from 1 to n inclusive. The factorial quickly becomes very large; 13! is too large to store as an integer on most computers, and 35! is too large for a floating-point variable. Your task is to find the rightmost non-zero digit of n!. (1<= n <= 100) For example, 5! = 1 * 2 * 3 * 4 * 5 = 120, so the rightmost non-zero digit of 5! is 2. Also, 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040, so the rightmost non-zero digit of 7! is 4.

Test cases

Input	Output
3	6
10	8

7. Write a C program that will allow a person to play a game of tic-tac-toe against the computer. Write the program in such a manner that the computer can be either the first or the second player. If the computer is the first player, let the first move be generated randomly. Write out the complete status of the game after each move. Have the computer acknowledge a win by either player when it occurs.

(OR)

Implement the program shown in Example 7.11 of Text book (Byron Gottfried). Modify it so that a sequence of craps games will be simulated automatically, in a non-interactive manner. Enter the total number of games as an input variable. Include within the program a counter that will determine the total number of wins. Use the program to simulate a large number of games (e.g., 1000). Estimate the probability of coming out ahead when playing multiple games of craps. This value, expressed as a decimal, is equal to the number of wins divided by the total number of games played. If the probability exceeds 0.500, it favours the player; otherwise it favours the house.

~~~~~~

If you try to solve problems yourself, then you will learn many things automatically. Spend few minutes and then enjoy the study.

~~~~~~