National University of Computer and Emerging Sciences



Lab Exercise 04

For

Object Oriented Programming Lab

|  |  |
| --- | --- |
| Course Instructor(s) | Mr. Usman Ghous |
| Lab Instructor(s) | Mr. Usman Ghous |
| Semester | Spring 2021 |

**FAST School of Computing**

# Instructions:

1. Make a word document with the naming convention “SECTION\_ LAB#\_ROLLNO” and put all your source code and snapshots of its output in it. Make sure your word file is formatted properly.
2. Plagiarism is strictly prohibited.
3. Do not discuss solutions with one another.

# Useful links

|  |
| --- |
| **Question#1** |

Write a function to recursively see if a number is prime or not, the function should return true when the base case has met, false otherwise.

|  |
| --- |
| **Question#2** |

Write a recursive code to find the factorial of a number. Then take two inputs a and b from user,  
and print the factorials of all the numbers in that range.

For example,

**Input a: 2  
Input b: 5  
Output:-  
Factorial of 2: 2  
Factorial of 3: 6  
Factorial of 4: 24  
Factorial of 5: 120**

**Question#3 (Recursion)**

Write a Recursivefunction to print a Fibonacci series up to N numbers  
Enter the number of elements: 10  
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34  
Note: Use minimum lines of code as possible to get max marks.

**Question#4 (Recursion)**

Write a function to recursively test if a string is palindrome or not, the function should return true when the base case has met, false otherwise. For example, string ‘aaa’ is a palindrome and ‘abab’ is not a palindrome. **Remember you need to provide a recursive solution.**

**Question#5 (Recursion)**

Write a recursive function to reverse a string. Write a recursive function to reverse the words in a string, i.e., “cat is running” becomes “running is cat”.

**Question#6 (Recursion)**

Write **recursive code** that computes all permutations of a string. For Example, if ourstring is abc, the permutations are: abc, bac, bca, acb, cab, cba. The following algorithm works this way:

1. Take out 'a' from "abc".

2. Now, we have ["bc","cb"].

3. Insert the 'a' into "bc", so that we have "abc", "bac", and "bca". In the same way, put 'a' into "cb", then we get "acb", "cab", and "cba".

**Question#7 (Recursion)**

Write a program to perform bubble sort on an array.  
1. Size and elements of array should be entered by user.  
2. Use recursive function to perform bubble sort  
Note: Use minimum lines of code as possible to get max marks.

**Question#8 (Recursion)**

Write a program to perform Insertion sort on an array.  
1. Size and elements of array should be entered by user.  
2. Use recursive function to perform Insertion sort  
Note: Use minimum lines of code as possible to get max marks.

**Question#9 (Recursion)**

The Tower of Hanoi is a mathematical puzzle. It consists of three poles and a number of disks of different sizes which can slide onto any poles. The puzzle starts with the disk in a neat stack in ascending order of size in one pole, the smallest at the top thus making a conical shape. The objective of the puzzle is to move all the disks from one pole (say ‘source pole’) to another pole (say ‘destination pole’) with the help of the third pole (say auxiliary pole).

The puzzle has the following two rules:  
      1. You can’t place a larger disk onto a smaller disk   
      2. Only one disk can be moved at a time.

Example:

Input: 3

Output:

Move disk 1 from rod A to rod B.

Move disk 2 from rod A to rod C.

Move disk 3 from rod A to rod D.

Move disk 2 from rod C to rod D.

Move disk 1 from rod B to rod D.

Input: 5

Output:

Move disk 1 from rod A to rod C.

Move disk 2 from rod A to rod D.

Move disk 3 from rod A to rod B.

Move disk 2 from rod D to rod B.

Move disk 1 from rod C to rod B.

Move disk 4 from rod A to rod C.

Move disk 5 from rod A to rod D.

Move disk 4 from rod C to rod D.

Move disk 1 from rod B to rod A.

Move disk 2 from rod B to rod C.

Move disk 3 from rod B to rod D.

Move disk 2 from rod C to rod D.

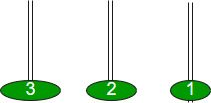
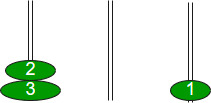
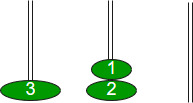
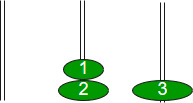
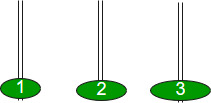
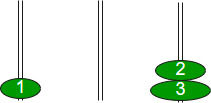
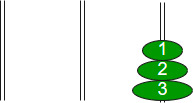
Move disk 1 from rod A to rod D.

Figure 8

Figure 7

Figure 6

Figure 4

Figure 5

Figure 3

Figure 1

Figure 2

**#10 (Recursion)**

Given the following three values, the task is to find the total number of maximum chocolates you can eat.

1. money: Money you have to buy chocolates.
2. price: Price of a chocolate
3. wrap: Number of wrappers to be returned for getting one extra chocolate.

It may be assumed that all given values are positive integers and greater than 1.  
**Examples:**

**Input:** money = 16, price = 2, wrap = 2

**Output:**  15

Price of a chocolate is 2. You can buy **8** chocolates from

amount 16. You can return 8 wrappers back and get **4** more.

chocolates. Then you can return 4 wrappers and get **2** more

chocolates. Finally, you can return 2 wrappers to get **1.**

more chocolate.

**Input:**  money = 15, price = 1, wrap = 3

**Output:**  22

We buy and eat 15 chocolates.

We return 15 wrappers and get 5 more chocolates.

We return 3 wrappers, get 1 chocolate and eat it

(keep 2 wrappers). Now we have 3 wrappers. Return

3 and get 1 more chocolate.

So total chocolates = 15 + 5 + 1 + 1

**Input:**  money = 20, price = 3, wrap = 5

**Output:**  7

A **naive method**is to continuously count the number of chocolates by returning wrappers until wrappers left didn’t become less than required to get a chocolate.

**Question#11 (Recursion)**

Suppose you have 8 chess queens......and a chess board.

* Can the queens be placed on the board so that no two queens are attacking each other.
* Two queens are not allowed in the same row...A picture containing checker

  Description automatically generated
* Two queens are not allowed in the same row, or in the same column...

A picture containing shoji, checker

Description automatically generated

* Two queens are not allowed in the same row, or in the same column, or along the same diagonal.

A picture containing shoji, building

Description automatically generated

* The number of queens, and the size of the board can vary.

**You have to write a program which tries to find a way to place N queens on an N x N chess board.**

**Algorithm:**

Place(k, i)

// Returns true if a queen can be placed

// in kth row and ith column. Otherwise it

// returns false. X[] is a global array

// whose first (k-1) values have been set.

// Abs( ) returns absolute value of r

{

for j := 1 to k-1 do

// Two in the same column

// or in the same diagonal

if ((x[j] == i) or

(abs(x[j] – i) = Abs(j – k)))

then return false;

return true;

}

**Algorithm nQueens(k, n) :**

// Using backtracking, this procedure prints all

// possible placements of n queens on an n×n

// chessboard so that they are nonattacking.

{

for i:= 1 to n do

{

if Place(k, i) then

{

x[k] = i;

if (k == n)

write (x[1:n]);

else

NQueens(k+1, n);

}

}

}

Input: 4 Queens on a 4x4 Board

Output:

Square

Description automatically generated