Discrete Structures Practical Lab

Note: From a computer science perspective, for better understanding of important discrete structures topics like Graphs, students must be taught from an implementation point of view. For the programs on recursion, students must print the intermediate steps, if possible. The computer language used for implementation would be C++.

1. Write a Program to create a SET **A** and determine the cardinality of SET for an input array of elements (repetition allowed) and perform the following operations on the SET:
    a) ismember (a, A): check whether an element belongs to set or not and return value as true/false.
    b) powerset(A): list all the elements of power set of A.
2. Create a class SET and take two sets as input from user to perform following SET Operations:
    a) Subset: Check whether one set is a subset of other or not.
    b) Union and Intersection of two Sets.
    c) Complement: Assume Universal Set as per the input elements from the user.
    d) Set Difference and Symmetric Difference between two SETS
    e) Cartesian Product of Sets.
3. Create a class RELATION, use Matrix notation to represent a relation. Include functions to check if a relation is reflexive, Symmetric, Anti-symmetric and Transitive. Write a Program to use this class.
4. Use the functions defined in Ques 3 to find check whether the given relation is:
    a) Equivalent, or
    b) Partial Order relation, or
    c) None
5. Write a Program to generate the Fibonacci Series using recursion.
6. Write a Program to implement Tower of Hanoi using recursion.
7. Write a Program to implement binary search using recursion.
8. Write a Program to implement Bubble Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.
9. Write a Program to implement Insertion Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.
10. Write a Program that generates all the permutations of a given set of digits, with or without repetition. (For example, if the given set is {1,2}, the permutations are 12 and 21). (One method is given in Liu)

11. Write a Program to calculate Permutation and Combination for an input value **n** and **r** using recursive formula of ${}^{n}C_{r}$ and ${}^{n}P_{r}$.
12. For any number **n**, write a program to list all the solutions of the equation $x_1 + x_2 + x_3 + \ldots + x_n = C$, where C is a constant (C<=10) and $x_1, x_2, x_3, \ldots, x_n$ are nonnegative integers using brute force strategy.
13. Write a Program to accept the truth values of variables **x** and **y**, and print the truth table of the following logical operations:
    a) Conjunction
    b) Disjunction
    c) Exclusive OR
    d) Conditional
    e) Bi-conditional
    f) Exclusive NOR
    g) Negation
    h) NAND
    i) NOR
14. Write a program to accept an input **n** from the user and graphically represent the values of **T(n)** where **n** varies from **0** to **n** for the recurrence relations. For e.g. T(n) = T(n-1) + n, T(0) = 1, T(n) = T(n-1) + n^2, T(0) =1, T(n) = 2*T(n)/2 + n, T(1)=1.
15. Write a Program to store a function (polynomial/exponential), and then evaluate the polynomial. (For example store **f(x) = 4n^3 + 2n + 9** in an array and for a given value of **n**, say **n = 5**, evaluate (i.e. compute the value of **f(5)**).
16. Write a Program to represent Graphs using the Adjacency Matrices and check if it is a complete graph.
17. Write a Program to accept a directed graph G and compute the in-degree and out-degree of each vertex.
18. Given a graph G, Write a Program to find the number of paths of length **n** between the source and destination entered by the user.
19. Given an adjacency matrix of a graph, write a program to check whether a given set of vertices $\{v_1, v_2, v_3 \ldots\ldots, v_k\}$ forms an Euler path / Euler Circuit (for circuit assume $v_k = v_1$).
20. Given a full **m-ary** tree with **i** internal vertices, Write a Program to find the number of leaf nodes.