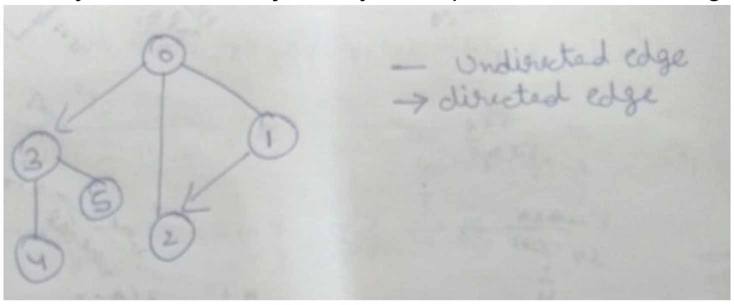
SOFTWARE CPC 4

MAX MARKS: 45

Section A
Basics of programming and OOP
(25 marks)

1) Define a graph. (2)
2) How does depth first and breadth first traversals works? (4)
3) Explain the following terms w.r.t graph theory: (7)
    a) Vertex and edges
    b) Simple graph
    c) NULL/empty graph
    d) Self edge
    e) Directed graph and Undirected graph
    f) Minimum spanning tree
    g) Cycles in a graph
4) Write the adjacency matrix and adjacency list representation of this graph: (4)



5) What is the time complexity of a recursive non-DP fibonacci function for calculating the nth fibonacci number. What is the complexity for the same if dynamic programming (Memoization) is introduced? (2)
6) Define a directed acyclic graph. Explain the topological sorting algorithm on a directed acyclic graph. (6)

Section B
Data structures and algorithms
(20 marks)

Note: Marks will be awarded for correct and efficient code. A well written code with a better time and space complexity will fetch you more marks.

1) Assume you're given a map of an area in the form of a 2-D matrix (N * M). The values of the matrix (A[i][j]) represent the amount of gold present at location (i, j). When you reach location (i, j) on the map, you can keep the gold at that location. Initially you're at (0,0) (top left) and you have to reach (n - 1, m - 1) on the map (bottom right). Out of all the possible paths that you can take, find the one that gives you maximum amount of gold and return the value. At any (i,j) you can either take a right (i, j + 1) or a down (i + 1, j).

   Function prototype: (5)

```
int gold_digger(int **arr, int n, int m)
{
        //Definition here
}
```

Example A =

1	4	6
2	5	2
2	6	9

```
Return value: 25.
Explanation: Let 'R' denote right movement and 'D' denote down movement then the
following sequence of moves:
```

RDDR

```
will fetch you 25 units of gold which also happens to be the maximum amount of gold
you can get following any sequence of moves.
```

2) Given two strings which are anagrams of each other, find the longest common subsequence of the two. A subsequence of a string S is string which can be obtained by deleting some characters from it while maintaining the order of the character. For example, S = "sample", few of the possible subsequences are "sple", "ape", "ampe", "sme" etc. A common subsequence of two strings is string that is a subsequence of both the strings. Your function must return the length of the longest such common subsequence.

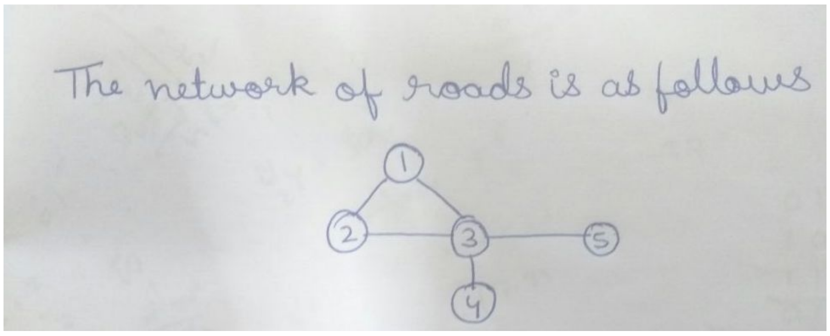
   Function prototype : (7)

```
int lcs(char s1[], char s2[])
{
        //Definition here
}
```

```
Example:        s1 = "bacge"
                s2 = "aecbg"
Return value: 3
Explanation: "acg" is the longest common subsequence of the two stings and has a
length of 3.
```

3) Tom lives in a city which is connected to other cities with bidirectional roads (may not be directly connected to all cities). He has friends in a few of the other cities and he likes to visit them once in while. He follows the following routine while visiting his K friends. He visits the first friend then goes back home, then visits second friend and goes back home. He repeats this until he has visited all his friends and finally back to his home. His vehicle consumes 1 unit of fuel for each road he takes from city x to city y. Write a **PROGRAM** that prints the minimum amount of fuel with which tom can visit all his friends in the above defined fashion and be back home. Tom resides in city numbered ‘1’.
Input Description:
First line has two integers N (number of cities), M (number of roads) and
K (number of friends tom has). Then M lines follow each with 2 integers a and b denoting a bidirectional road b/w city a and b (a <= N and b <= N). Then K space separated integers that denote the cities in which the K friends reside (K[i] <= N).
Your program must print the minimum amount of fuel required. (8)

```
Example:
       5 5 3
       1 2
       1 3
       2 3
       3 4
       3 5            (Roads data ends here)
       3 5 2          (Cities where tom’s friends live)
Output: 8
Explanation:
```



```
To visit his first friend, he can take the following route:
       1->3 and then the same route back i.e. 3->1. This consumes 2 units of fuel
To visit his second friend, he can take the following route:
       1->3->5 and then same route back so it consumes 2 + 2 = 4 units of fuel
To visit his 3rd friend, he can take 1->2 and same route to back home so 2 units of
fuel is consumed.
Therefore the total amount of fuel consumed is 2 + 4 + 2 = 8 units and your program
should therefore print 8.
```

All the best :)