SOFTWARE

MAX MARKS: 50

Note: For all output questions, assume that standard gcc compiler is used.

Section A

Basics of programming

(12 marks)

1) Predict the output for the following program. (+1, -0.5)

```
#include<stdio.h>
int main()
{
    union var
    {int a, b;};
    union var v;
    v.a=10;
    v.b=20;
    printf("%d\n", v.a);
    return 0;
}
```

A) 10 B) 20 C) Garbage value D) Compilation error

2) Which of the following is not a logical operator? (+1, -0.5)

A) && B) ! C) & D) ||

3) Predict the output of the following program. (+1, -0.5)

```
#include<stdio.h>
int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

A) 2, 1, 15 B) 2, 2, 15 C) 3, 2, 15 D) 2, 3, 15

4) Predict the output of the following program: (+1, -0.5)

```
#include<stdio.h>
#define CUBE(x) (x*x*x)
int main()
{
    int a, b=3;
    a = CUBE(b++);
    printf("%d, %d\n", a, b);
    return 0;
}
```

A) 9, 4 B) 27, 4 C) 27, 6 D) 27, 9

5) Which of the following has the best worst case complexity- (+1, -0.5)
A) Insertion sort B) Selection sort C) Quicksort D) Mergesort

6) Prove that the time complexity of binary search in O(log(N)). (+3)
7) What is a weighted graph? Explain the shortest path algorithm on a weighted graph. (+4)

Section B
Data structures and algorithms
(25 marks)

1) Write a **PROGRAM** that performs range sum query in an efficient manner. Range sum query involves printing the sum of the array elements from **[i...j]** i.e. given two integer indices **'i'** and **'j'**, print the sum of elements **a[i]+a[i+]...+a[j]**. Input specification is as follows:

    First line contains an integer **N** denoting the size of array
    Second line contains **N** space separated integers denoting the array itself.
    Next line contains an integer **Q** denoting the number of range queries to follow.
    Next **Q** lines contain two integers each, **i** and **j**.
    Your program must output the sum of elements for each of the **Q** queries.: (5)

```
Example input: 5              (N)
               1 2 3 4 5      (Array)
               2              (Q)
               0 2            (i, j)
               1 3            (i, j)

Expected Output:      6       (Sum of elements from index 0 to 2)
                      9       (Sum of elements from index 1 to 3)
```

2) Write a function that takes as input head to a linked list and an integer **K** (non negative integer) and returns a modified list by rotating the original list by **K** places. Function prototype is as follows:
    (5)

```
node * rotate_list(node *head)
{
        //Definition here
}
```

Where node is defined as follows

```
struct node
{
        int data;
        node *next;
}
```

```
Example input list: 1->2->3->4->NULL, K = 2
Expected modified list: 3->4->1->2->NULL
```

3) Write a function to compute the next greater element for each element of an array. The next greater element for an element A[i] is defined as the closed right element

that is just greater than it. See sample output for clarification. Your function takes as input the array and it's size. Print the next greater element for each A[i] and print '-1' if it does not exist. Function prototype:

(5)

```
        void next_greater(int arr[], int size)
        {
                //Definition here
        }
Example input:          6  4  7  1  5  2  3
Expected output:        7  7 -1  5 -1  3 -1
```

4) Write a function that takes as input the root of a binary tree and prints all it's elements in a zigzag level wise order. See example for understanding the order. The arguments to your function is the root of the tree.:

(5)

```
        int zigzag(treenode *root)
        {
                //Definition here
        }
```

Where treenode is defined as follows

```
                struct treenode
                {
                        int data;
                        treenode *left;
                        Treenode *right;
                }
Example tree:                      5
                                  / \
                                 3   6
                                /   / \
                               1   5   7
Expected output: 5 6 3 1 5 7
(Level 1 is printed left to right, level 2 is printed right to left, level 3 is
printed left to right and so on)
```

5) Write a function that takes as argument an array, it's size and an integer value **V** and returns '1' if a subset of the elements can be chosen in such a way that they sum up to **V**. Return '0' otherwise. Function prototype is as follows: (5)

```
        int subset_sum(int arr[], int size, int v)
        {
                //Definition here
        }
Example input: 3 5 1 4 6, v = 8
Return value:  1 ({3,5} or {3, 1, 4} sum up to 8. )

Example input: 3 5 1 4 6, v = 17
Return value: 0 (No subset of the elements sum up to 17)
```

Section C
Object oriented programming
(13 marks)

1) Explain briefly any four features of object oriented programming. (4)
2) Explain the different access controls present in classes. How is the ‘protected’ specifier different from ‘private’ specifier. (2)
3) How is a pure virtual function defined? What are the characteristics of an abstract class? (2)
4) Is there any way to allow external functions to access a class’s private members? If so, how? (2)
5) Explain the reason behind passing the argument by reference and not by value to a copy constructor. (1)
6) Predict the ouptut: (2)

```
#include<iostream>
using namespace std;
class Base1 {
 public:
     Base1()
     { cout << " Base1's constructor called" << endl;  }
};

class Base2 {
 public:
     Base2()
     { cout << "Base2's constructor called" << endl;  }
};

class Derived: public Base1, public Base2 {
   public:
     Derived()
     {  cout << "Derived's constructor called" << endl;  }
};

int main()
{
   Derived d;
   return 0;
}
```

ALL THE BEST :)