Question 1 [30 Marks]

BST and Logical

1. Assume a binary tree is stored in an array called treeNodes, which has a capacity of 100 array elements, with array index 0 not being used. If the binary search Tree currently contains 86keys, answer the following questions. [1+2+2+1+1+1=8 marks]

   a. Is treeNodes[44]a leaf node? [1 Mark]

   Yes, as its children would be 88 and 89 which are unused 1

   b. Does treeNodes[43] have only one child node? explain. [2 Marks]

   Yes, as its children would be 86 and 87 and 87 is unused 2

   c. Is the subtree rooted at treeNodes[8] a full binary tree? Explain.[2 Marks]

   Yes, as it has all children and their children within 86. 2

   d. How many levels are there in the subtree rooted at treeNodes[8]including node itself? [1Marks]

   4 1

   e. Including the root node(treeNodes[1])how many levels are there in the binary Search Tree? [1 Marks]

   6 0.

   f. How many leaf nodes are there? [1 Marks]

   32 0.

2. Write a C++ recursive function called "CountLeaf(Node *root)". This function will count all the leaf nodes in a binary tree. Marks will be awarded according to optimality of your code and zero marks will be awarded for non-recursive solution. [7 Marks]

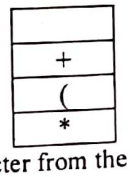
```
int CountLeaf ( Node * root )
{
    int count = 0
    CountLeafn ( Node * root, count) ;
    return count;
}

id Countleafn ( Node *root, int &num )
{
    if( root != null)
    Countleafn ( root -> left , num) ;
    num ++;
    Countleafn ( root -> right , num) ;
}
```

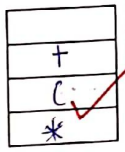
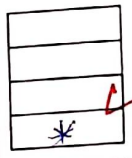
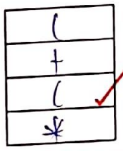
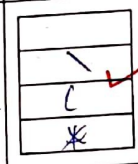
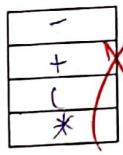
3. Consider the usual algorithm that converts ***infix to postfix***. Suppose you have already traversed 10 characters during conversion and the stack contains following symbols

A + B + C

AB + +



a. Now you are going to read 11th character from the input. Draw the stack for each case when 11th symbol is as follows: [5 Marks]

1.	A number		2.	A right Parenthesis	
3.	A left parenthesis		4.	A division Sign	
5.	A minus sign				

4. Assume there exists a singly linked list with n number of nodes and a head pointer to its first node (**No tail Pointer available**). Write a function in C++ named ***void moved_to_start(int value)*** where node with **value** in its data is moved to front of the list and rest of the list remains unchanged. (***Note: you are not allowed to simply swap values. You must move the complete node to a different position. Assume there are no duplicate values in the list***). Address all possible cases. [10 Marks]

```
void moved_to_start(int v)
{
    Node* temp = head, prev = null;
    while (temp.data != v && temp != null)
    {
        prev = temp
        temp = temp -> next;
    }
    if (temp.data == v)
    {
        prev.next = temp.next;
        temp.next = head;
        head = temp;
    }
}
```

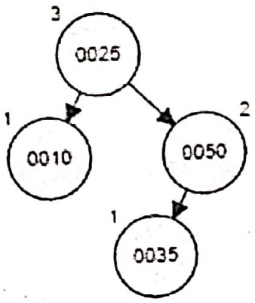
National University of Computer and Emerging Sciences

Question 2 [20 Marks]

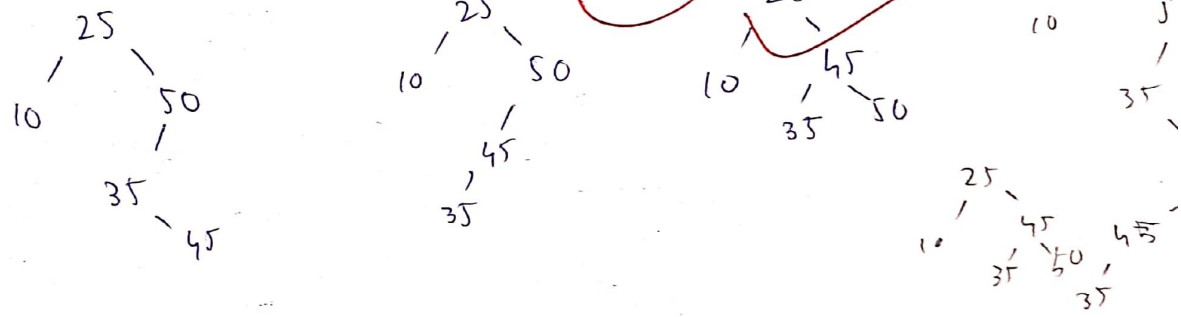
AVL

1. Perform the following operations in order, drawing the result after each operation and using it as the starting point for the next operation. You only need to show the result of the operation but showing your work will allow partial credit in case of error. [10 Marks]

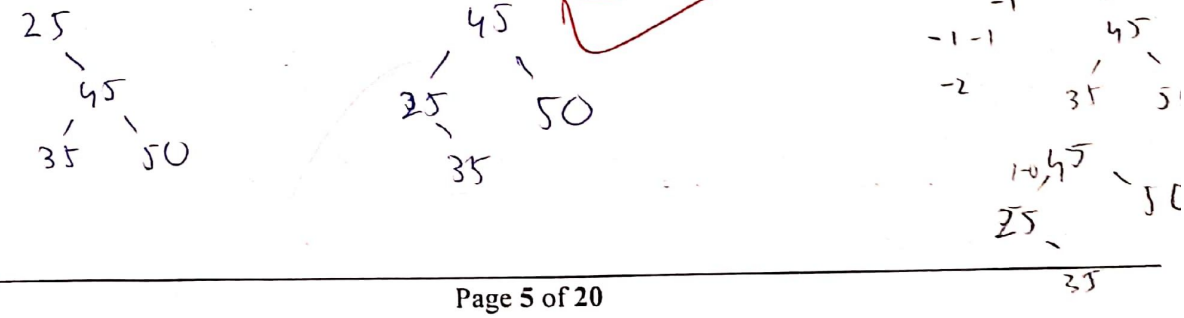
   Circle the result of each operation so we can distinguish it from intermediate work.

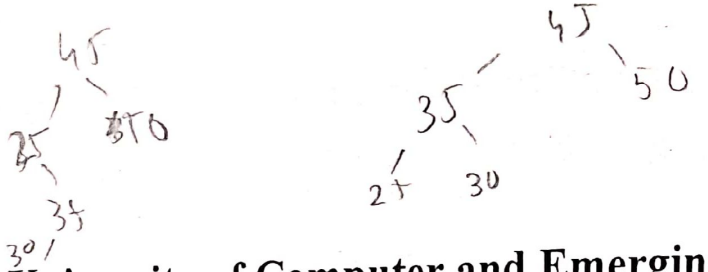


a) Insert 45 [3 Marks]

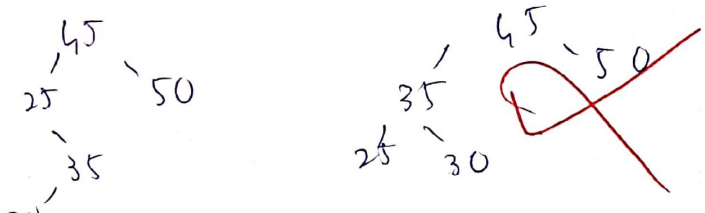


b) Delete 10 [4 Marks]

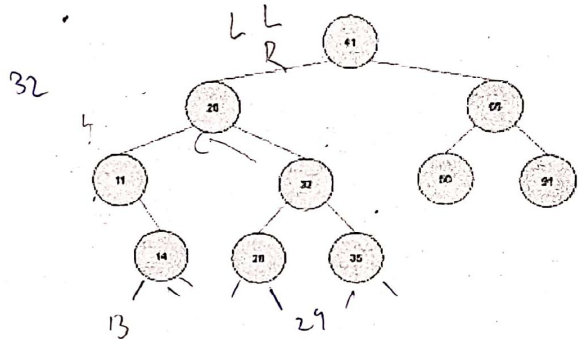




:) Insert 30 [3 Marks]



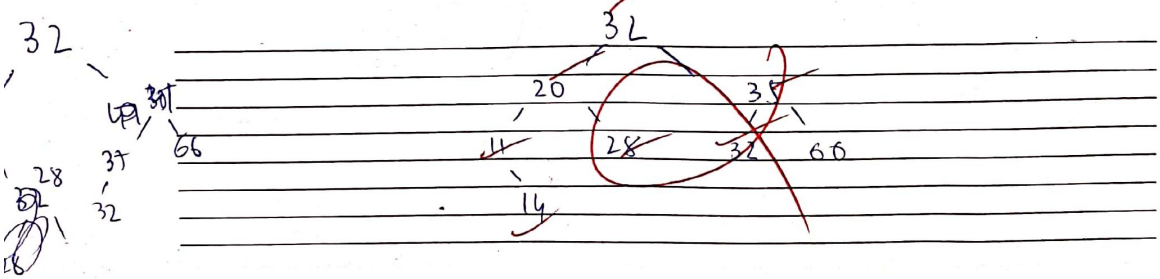
2. Suppose we had the following AVL Tree. We are interested in figuring out what value would be necessary to insert into the tree, such that after any AVL rotation(s), the node with the value 32 become the root of the whole tree. [3+5=8 marks]



a. What single value could we insert into this AVL tree using an AVL insert operation that would cause the root of the tree to become the node with the key 32? There may be multiple correct answers, but you should pick one. Recall, we don't allow duplicates in these trees. [3 Marks]

33, this will cause a RL violation on the root 41

b. Draw the final tree after the insertion happens and any rotations occur. [5 Marks]



Question 3 [25 Marks]

ck Tree

sider a Red Black tree which is initially empty. A teacher wants to store the score of their class in RB tree. y want to store students' roll numbers and course score for each person. RB will be generated on the base ourse score.

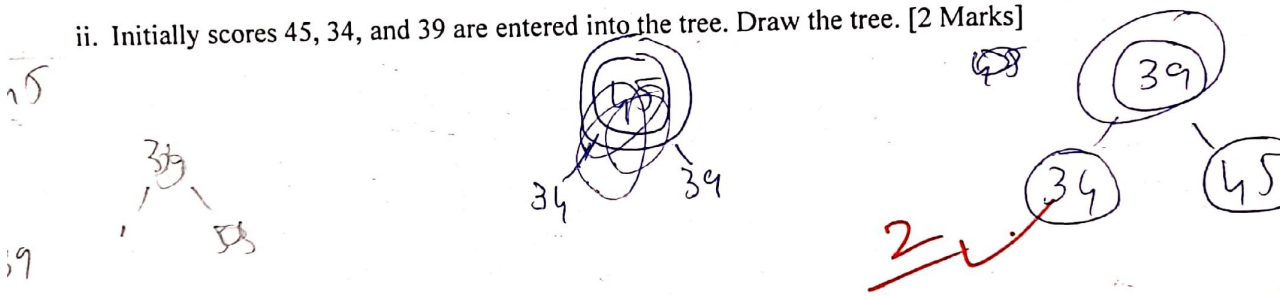
i. Write down the code in C++ for the structure of each node in the above defined RB tree [1 Mark]

```
struct RB_n {
    int roll_no;
    int score;
    bool is_red;
    RB_n * left;
    RB_n * right;
```

✓1

)TE: for next parts draw RB tree using scores only and don't show Null Leaves.
dicate colours of nodes as: single circle = Red node.
wo concentric circles=Black node

ii. Initially scores 45, 34, and 39 are entered into the tree. Draw the tree. [2 Marks]

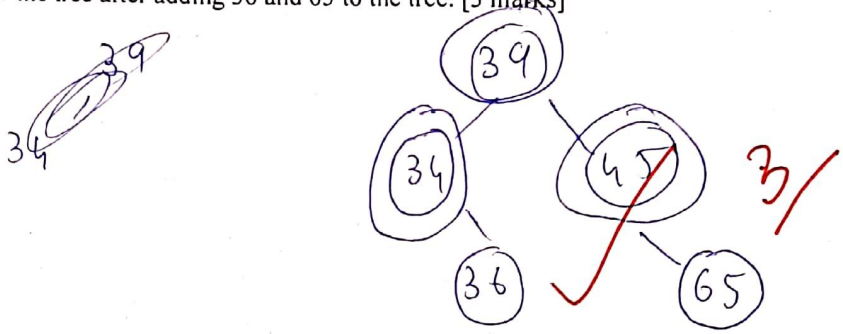


iii. What is the height of the tree? And what rotation was applied to tree (if any. Write NILL if none was applied) [1 Mark]

h = 1

r = RL rotation ✓ 1

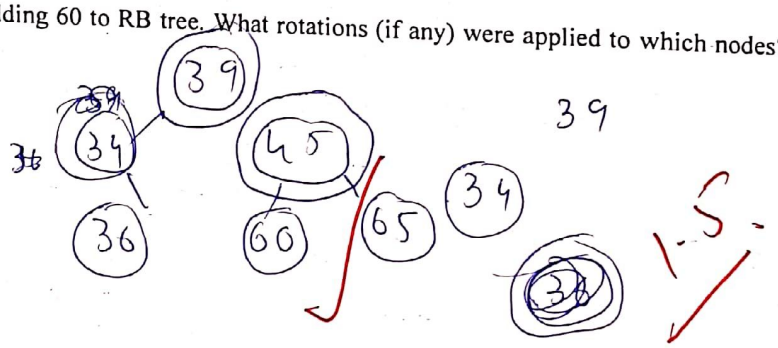
iv. Redraw the tree after adding 36 and 65 to the tree. [3 marks]



v. What is the colour of NULL leaf nodes in RB trees? [1 Mark]

Black

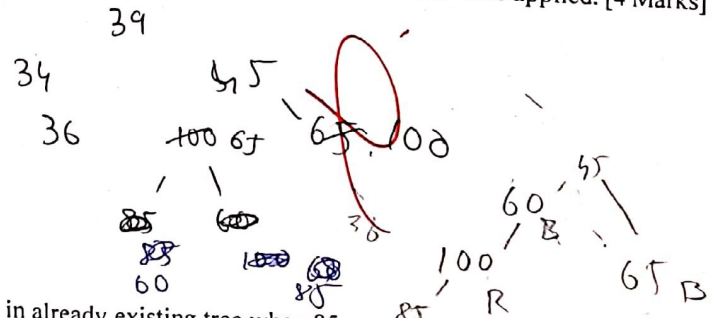
vi. Redraw the tree after adding 60 to RB tree. What rotations (if any) were applied to which nodes? [2 Marks]



vii. Add 100 and 85 respectively to updated tree and redraw. Indicate any rotations that were applied. [4 Marks]

for black node

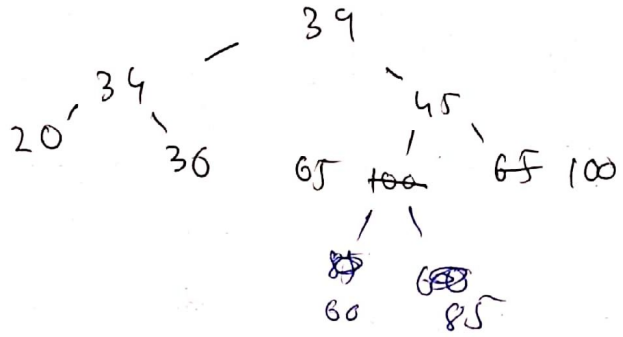
for Red node



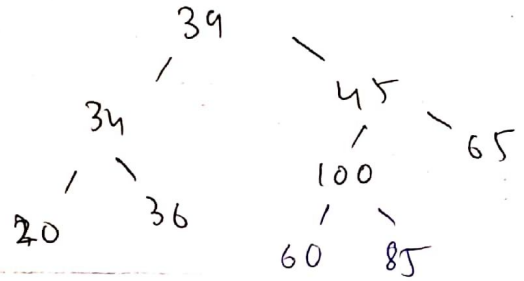
viii. Which nodes changed colour in already existing tree when 85 was added? [1 Mark]

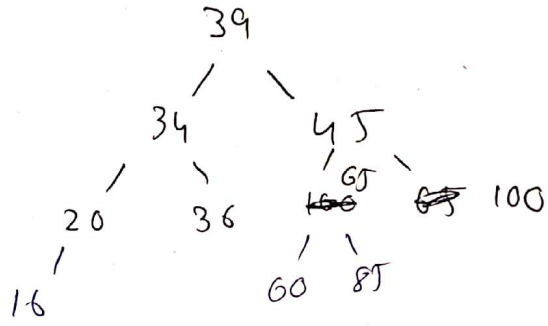
60, 100

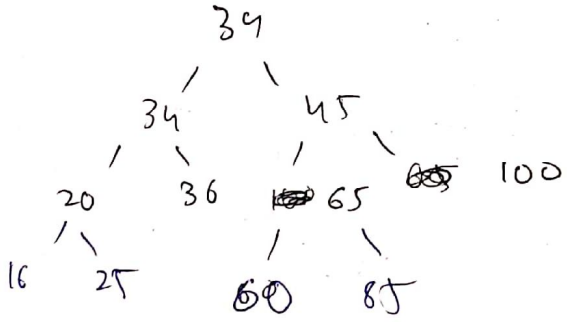
ix.Insert 20 and redraw the tree. [2 Marks]



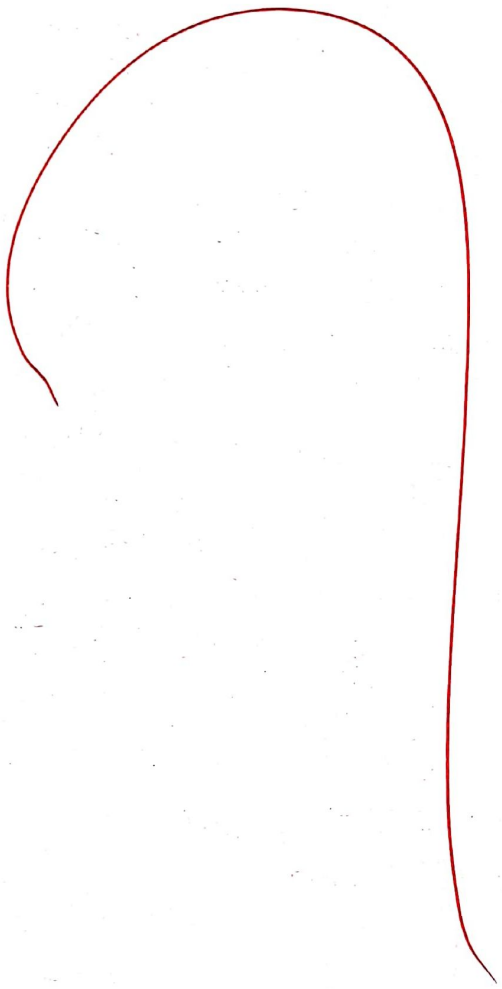
x.Add 22,16 and 25. Identify for each insertion if rotations are required? Identify the rotations and recolouring and apply, then redraw the tree at each step of process. [5 Marks]







xi. A student suddenly decided to drop the course. His score is 45. Remove the student's record from the tree and redraw the tree. Also identify and perform any rotations occur. Redraw the final tree.[3 Marks]



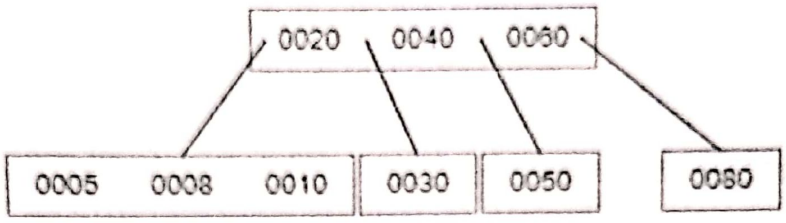
Question 4 [20 Marks]

B Trees

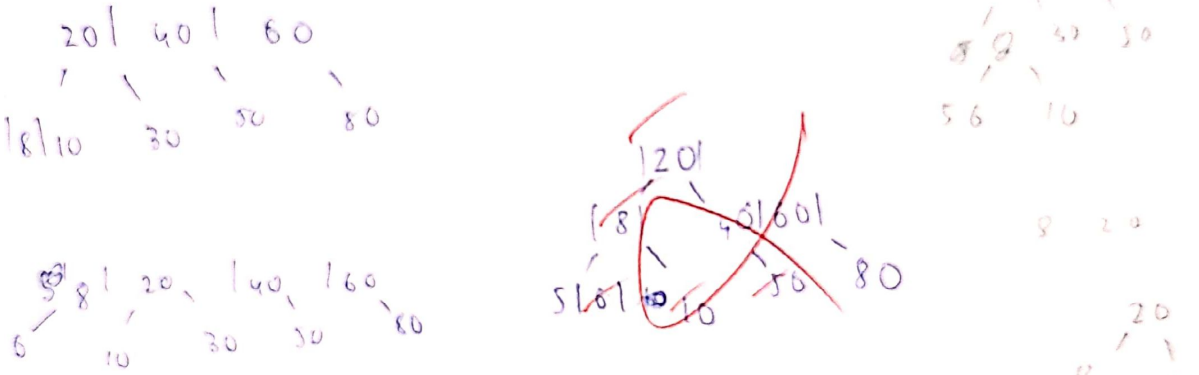
1. Perform the following operations in order, drawing the result after each operation and using it as th starting point for the next operation. You only need to show the result of the operation but showing you work will allow partial credit in case of error.

   Circle the result of each operation so we can distinguish it from intermediate work.

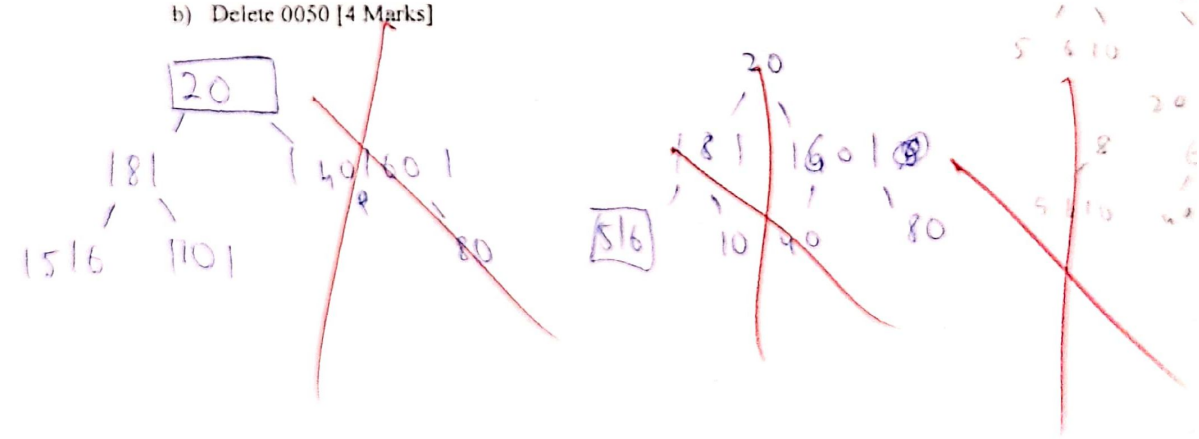
Assuming



a) Insert 006 [3 Marks]



b) Delete 0050 [4 Marks]



c) True or false and explain your answer briefly: If the root of a B tree is full, then an insert operation definitely will increase the height of the tree. [2 Marks]

if key is entering at root
True ∧ as B tree grows upward not downwards

else
false

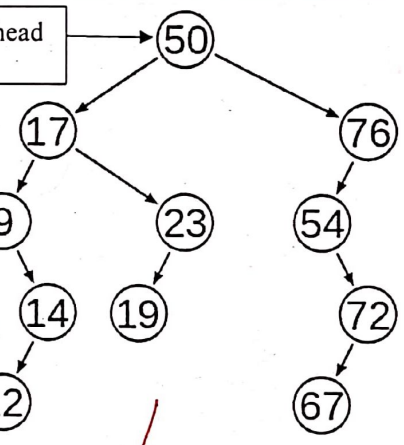
d) Consider a B-Tree as defined in lecture and the textbook where m = 64 and the number of nodes is n. Describe an $O(\log_m n)$ algorithm for returning the 4th smallest element of the tree. You may assume the root is not a leaf. Don't write the code. [4]

You will go to the left+most child $\log_m$ times becuz each time you are bypassing that many values and n becuz you will need a loop to reach the corr key in a node

2

2. What will be the output of following code for given binary search tree. [7 Marks]

```
int count = 0;
Node* element(Node* root, int& k)
{
    if (root == NULL)
        return NULL;
    Node* left =  element(root->left, k);
    if (left != NULL)
        return left;
    count++;
    if (count == k)
        return root;
    return  element(root->right, k);
}
void show()
{
   Int arr[4]={6,4,2,8};
    for( int i=0; i<4; i++)
        cout<< element(head, a[i])<<endl;
}
Int main()
{
  Show();
}
```

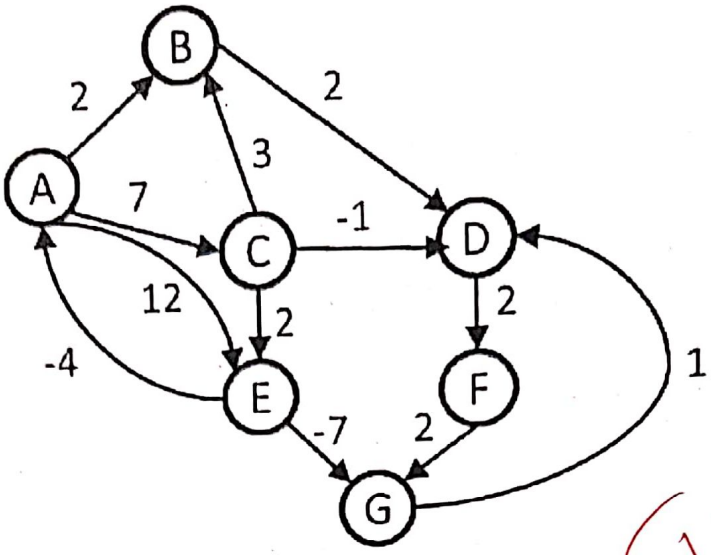


Output: (TP) Node 7 , 9

Question 5 [30 Marks]

Graphs

1. Consider the following directed, weighted graph:



	A	B	C	D	E	F	G
A	0	2	7	∞	12	∞	∞
B	∞	0	∞	2	∞	∞	∞
C	∞	3	0	-1	1	∞	∞
D	∞	∞	∞	0	∞	2	∞
E	-4	∞	∞	∞	0	∞	-7
F	∞	∞	∞	∞	∞	0	2
G	∞	∞	∞	7	∞	∞	0

a) Even though the graph has negative weight edges, step through Floyd algorithm to calculate supposedly shortest paths from every vertex to every other vertex. Show your steps in the table below.[10 Marks]

A 0 2 7 ∞ 12 ∞ ∞

E → A → B = -2
E → A → C = 3
A → B → D = 4
E → A → B → D = 0

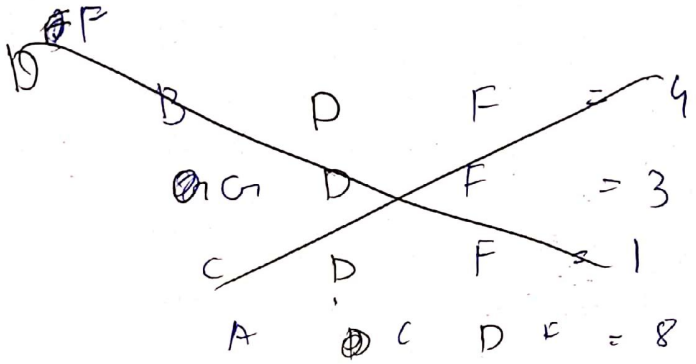
D^C
A C E = 9

D^D
A → B → D → F = 0
C → D → F = 1
B → D → F = 4
E → A → B → D → F = 2
G → D → F = 3

D^E

C E $G = -7$.

A $G = 5$



D^F

D F $G = 4$

B D F $G = 6$

C D F $G = 3$

A C D F $G = 2$

D^G

E $D = -6$

F $D = 3$

Final Matrix:

	A	B	C	D	E	F	G
A	0	2	7	4	9	6	5
B	2	0	2	2	2	4	6
C	-2	3	0	-1	2	1	-5
D	2	2	2	0	2	2	4
E	-4	-2	3	0	0	2	
F	2	2	2	3	2	0	2
G	2	2	2	1	2	3	0

b) Floyd algorithm found the wrong path to some of the vertices. For just the vertices where the wrong path was computed, indicate both the path that was computed and the correct path. [3 Marks]

c) What single edge could be removed from the graph such that Floyd algorithm would happen to compute correct answers for all vertices in the remaining graph? [2 Marks]

Edge (A, E)

d) Does Dijkstra compute the all-pair shortest path? If no, why? If yes, which one is better Dijkstra or Floyd? Why? [5 Marks]

Dijkstra can but it requires multiple calls So the complexity is extremly high compared to floyd. Furthermore floyd stores it results compares to Dijkstra

2. Assume that an undirected graph is stored in an adjacency matrix. Complete the method countEdges in the Graph class that returns the number of edges in the Graph. [5 Marks]

Note: Marks will be awarded according to optimality of your code.

int countEdges() { // computes and returns number of edges

}

```
int count = 0
for (int i = 0 ; i < n ; i++)
{
    for (int j = 0 ; j < n ; j++)
    {
        if (matrix[i][j] = 1)
            count++
    }
}
```

3. Given an algorithm for minimum spinning tree. Select the best option for blanks. [5 Marks]

```
1.void primMST(int graph[V][V])
2.{
3.   int parent[V];
4.   int key[V];
5.   bool mstSet[V];
6.   for (int i = 0; i < V; i++)
7.     key[i] = INT_MAX, mstSet[i] = false;

8.   key[0] = 0;
9.   parent[0] = _____;
10.  for (int count = 0; count < V - 1; count++) {
11.    // Pick the minimum key vertex from the
12.    // set of vertices not yet included in MST
13.    int u = minKey(key, mstSet);
14.    mstSet[u] = true;
15.
16.    for (int v = 0; v < V; v++)
17.      if (graph[u][v] && mstSet[v] == false
18.        && _____)
19.        parent[v] = u;
20.        key[v] = _____;
   }
}
```

a. Line 9:
   a) 0
   b) INT_MAX
   c) -1
   d) None

b. Line 18:
   a) graph[u][v] < mstSet[v]
   b) graph[u][v] < parent[v]
   c) graph[u][v] > key[v]
   d) graph[u][v] < key[v]

c. Line 20:
   a) u
   b) graph[u]
   c) graph[u][v]
   d) parent [v]