KIIT Deemed to be University
Online End Semester Examination(Autumn Semester-2020) Solution

Subject Name & Code: DSA (CS 2001)Regular **Applicable to Courses: B.Tech**

Full Marks=50 **Time:2 Hours**

SECTION-A(Answer All Questions. Each question carries 2 Marks)

Time:30 Minutes **(7×2=14 Marks)**

Question No	Question Type (MCQ/SAT)	Question	CO Mapping	Answer Key (For MCQ Questions only)

Q.No:1	MCQ	What is the time complexity of following code:
		int a = 0, b = 0;
		for (i = 0; i < N; i=i*2) {
		a = a + rand();
		}
		for (j = 0; j < M; j++) {
		b = b + rand();
		}
		
		(A) O(N * M) time
		(B) O(N + M) time
		(C) O(log N+ M) time
		(D) O(N * M) time
			CO2	(C)

	MCQ	What is the time complexity of following code:
		int i, j, k = 0;
		for (i = n/ 2; i <= n; i++) {
		for (j = 2; j <= n; j = j * 2)
		{
		k = k + n / 2;
		}
		}
		(A) O(n)
		(B) O(nLogn)
		(C) O(n^2)
		(D) O(n^2Logn)
			CO2	(B)

	MCQ	What is the time complexity of following code: int a = 0, i = m;	CO2	(D)

		while (i > 0) {
		a += i;
		i /= 2;
		}
		(A) O(m)
		(B) O(Sqrt(m))
		(C) O(m / 2)
		(D) O(log m)		

	MCQ	What is the time complexity of following code:
		int a = 0;
		for (i = 0; i < N; i++) {
		for (j = N; j > i; j--) {
		a = a + i + j;
		}
		}
		(A) O(N)
		(B) O(N*log(N))
		(C) O(N * Sqrt(N))
		(D) O(N*N)
			CO2	(D)

Q.No:2	MCQ	A binary search tree is generated by inserting integer values in order the order: 60, 25, 65, 15, 30, 55, 80, 10, 20, 40, 75, 35. Find the number of nodes in the left subtree, right subtree of the root and the height of the tree respectively. (A) (6, 5, 3) (B) (7, 4, 4) (C) (8, 3, 4) (D) (5, 6, 3)	CO 4	(C)
	MCQ	Suppose the following numbers are entered to construct a binary search tree. 100, 80, 200, 75, 84, 22, 63, 15, 7 Further, the tree is converted into a one way inorder threaded binary tree. How many threads will be present in the tree? (A) 4 (B) 5 (C) 7 (D) 8	CO 4	(B)

	MCQ	Suppose the following numbers are entered to construct a binary search tree. 80, 100, 75, 22, 63, 15, 7, 84, 200 Further, the tree is converted into a one way inorder threaded binary tree. How many threads will be present in the tree? (A) 4 (B) 7 (C) 5 (D) 8	CO 4	(C)
	MCQ	Suppose the following numbers are entered to construct a binary search tree. 22, 15, 20, 18, 75, 60, 50, 10 Further, the tree is converted into a one way inorder threaded binary tree. How many threads will be present in the tree? (A) 4 (B) 7 (C) 8 (D) 5	CO 4	(D)
Q.No:3	**MCQ**	Evaluate the following prefix expression. +, * ,2, +, /, 14, 2, 5, 1 (A) 25 (B) 24 (C) 23 (D) Fractional Value	CO1,CO 4	(A)
	MCQ	Evaluate the following postfix expression. 1, 4, 18, 6, /, 3, +, +, 5, /, + (A) 2 (B) 3 (C) 4 (D) 5		(B)
	MCQ	Evaluate the following postfix expression. 4, 3, 6, 3, *, 12, -, *, + (A) 22 (B) 18 (C) 25 (D) 3		(A)

	MCQ	Evaluate the following prefix expression. *, -, +, 4, 3, 5, /, +, 2, 4, 5 (A) 14 (B) 21 (C) 19 (D) 12/5		(D)

Q.No:4	**MCQ**	Consider the following code: void fun(struct node * start) { if (start==NULL) return;
		if(start->next!=NULL)
		fun(start->next->next);
		printf("%d",start->data);
		For a linked list with following data input to the above code, what will be the output of the code? 11->15->25->50->87->23?
		(A) 23 50 15
		(B) 87 25 11
		(C) 11 15 25 50 87 23
		(D) 11 25 87
			CO3	(B)

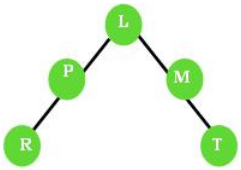
	MCQ	Consider a double circular linked list. Let P points to the start node of the linked list. Then the following code snippet will delete_____ node? [prev and next represent the previous and next pointer respectively] p->prev->prev->prev->next =p->prev p->prev->prev=p->prev->prev->prev (A) Last (B) Node before the last (C) First (D) Second	CO3	(B)

	MCQ	Consider the following function applied to a single linked list with odd no. of nodes :
		struct node * fun ( )
		{
		struct node *p, *q;
		p=q=start;// start points to the first node of the list
			CO3	(C)

		while(q!=NULL && q->next!=NULL)
		{
		q=q->next->next;
		p=p->next;
		}
		retrun p;
		}
		The code will return _____ of the list.
		(A) Last node
		(B) Node before the last node
		(C) Middle node
		(D) None of these		

	MCQ	Consider the following function applied to a single linked list with odd no. of nodes :
		void fun (struct node * start)
		{
		if(start!=NULL)
		printf("%d", start->data);
		fun(start->next);
		printf("%d",start->data);
		}
		The code will print the list___.
		(A) Two times in forward direction
		(B) One time forward direction and one time backward direction
		(C) Two time in backward direction
		(D) None of these
			CO3	(B)

Q.No:5	**MCQ**	With the following set of traversal sequences together, how many binary trees can be identified? Preorder : ABDEFCGHJLK Postorder: DFEBGLJKHCA (A) One (B) Two (C) Three (D) Four	CO4	(D)

	MCQ	With the following set of traversal sequences together, what is the equivalent postorder traversal sequence? Preorder : abcdefghijk Inorder: bacdfehgjki (A) bfhkjigedca (B) bfhkjigedac (C) bfhkijgedca (D) bhfkjigedca	CO4	(A)
	MCQ	What is the equivalent postfix expression for the prefix expression : ++ac*d-e/+fgh? (A) acd*+efg+h/+- (B) acd*+efg+h/-+ (C) acd*+feg+h/-+ (D) acd+*efg+h/-+	CO4	(B)
	MCQ	What is the equivalent prefix expression for the postfix expression : acd*+efg+h/-+? (A) ++ac*de-/+fgh (B) ++ac*d-e+/fgh (C) ++ac*d-e/+fgh (D) ++ac*d-e/+fgh	CO4	(C)
Q.No:6	**MCQ**	Figure below is a balanced binary tree. If a node inserted as child of the node R, how many nodes will become unbalanced?  (A) 2 (B) 1 (C) 3 (D) 0	CO4, CO1,CO 6	(B)
	MCQ	Maximum number of nodes present at any level of a tree is? A. n B. 2^n C. n+1 D. 2n	CO4, CO1,CO 6	(B)
	MCQ	What is the number of	CO4,	(C)

		edges present in a complete graph having n vertices? A. (n*(n+1))/2 B. n C. (n*(n-1))/2 D. nC2	CO1,CO 6	
	MCQ	A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place? (A) 3 (B) 4 (C) 5 (D) 6	CO4, CO1,CO 6	**(C)**
Q.No:7	MCQ	Given the following input (122, 634, 1976, 679, 989, 571, 6773, 2399) and the hash function x mod 10, which of the following statements are true? (A) 679, 989, 2399 are having collision (B) 571,1976 are having collision. (C) All elements hash to the same value (D) Each element hashes to a different value	CO5,CO 6	(A) 679, 989, 2399 are having collision
	MCQ	Suppose we have a O(n) time algorithm that finds median of an unsorted array. Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the worst case time complexity of this modified QuickSort. (A) O(n^2 Logn) (B) O(n^2) (C) O(n Logn Logn) (D) O(nLogn)	CO5,CO 6	(D)
	MCQ	Which of the following sorting algorithms in its typical implementation gives best performance	CO5,CO 6	(C) Insertion Sort

		when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced). (A) Quick Sort (B) Heap Sort (C) Insertion Sort (D) Merge Sort		
	MCQ	For merging two sorted lists of size m and n into sorted list of size m+n, no. of comparisons required ----------? a) O(m) b) O(n) c) O(m+n) d) O(logm + logn)	CO5,CO 6	(C)

SECTION-B
(Answer Any Three Questions. Each Question carries 12 Marks)

Time: 1 Hour and 30 Minutes **(3×12=36 Marks)**

Note: This solution is only a sample solution. Students could have written variation/ different code. So, marks should be awarded judiciously.

Question No	Question	CO Mapping
Q.No:8	What is the difference between a single linked list and double linked list representation? Write a C function to rotate a double linked list anti clock wise. Given a double linked list, rotate the linked list counter-clockwise by k nodes. Where k is a given positive integer. For example, if the given linked list is A->B->C->D->E->F and k is 3, the list should be modified to D->E->F->A->B->C. Assume that k is smaller than the count of nodes in linked list. [12]	**CO-3**

Evaluation Scheme:
Difference: [4 Mark]
Algorithm: [8 Mark]

Solution:

Singly Linked List	Doubly Linked List
In a singly linked list, the node contains some data and a pointer to the next node in the list.	In a doubly linked list, the node contains some data and a pointer to the next as well as the previous node in the list.
It uses less memory per node (one pointer).	It uses more memory per node (two pointers).

It can be traversed only in the forward direction.	It can be traversed in both directions.
Less efficient access to elements.	More efficient access to elements.

Declaration:
struct Node {
int data;
struct Node *next;
};
Draw the Structure
	Declaration:
	struct Node {
	int data;
	struct Node* prev;
	struct Node* next;
	};
	Draw the Structure

```
struct Node {
      char data;
      struct Node* prev;
      struct Node* next;
};


void rotate(struct Node** head, int k) {
     if (k == 0)
          return;
     struct Node *curr, *newHead;
     curr = *head;
     int count = 1;
     while (count < k && curr != NULL) {
          curr = curr->next;
          count++;
     }
     if (curr == NULL) return;
     newHead = curr;
     while (curr->next != NULL)
          curr = curr->next;
     curr->next = *head;
     (*head)->prev = curr;
     *head = newHead->next;
     (*head)->prev = NULL;
     neweHead->next = NULL;
}
```

Question No	**Question**	**CO Mapping**
Q.No:8	What is the difference between an array and linked list representation? Write function to reverse a single list starting with k node followed by k+1, k+2, …The function should reverse first k nodes in the list, then reverse next k+1 nodes in the list, then reverse next k+2 node in the list, and like this, it will continue till the end of the list.[12]	**CO-3**

Evaluation Scheme:
Difference: [4 Mark]
Algorithm: [8 Mark]

Solution:

Comparison between Array and Single linked list:
1. An array is the data structure that contains a collection of similar type data elements whereas the Linked list is considered as a non-primitive data structure contains a collection of unordered linked elements known as nodes.
2. In the array the elements belong to indexes, i.e., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket while in a linked list though, you have to start from the head and work your way through until you get to the fourth element.
3. Accessing an element in an array is fast, while Linked list takes linear time, so it is quite a bit slower.
4. Operations like insertion and deletion in arrays consume a lot of time. On the other hand, the performance of these operations in Linked lists are fast.
5. Arrays are of fixed size. In contrast, Linked lists are dynamic and flexible and can expand and contract its size.
6. In an array, memory is assigned during compile time while in a Linked list it is allocated during execution or runtime.
7. Elements are stored consecutively in arrays whereas it is stored randomly in Linked lists.
8. The requirement of memory is less due to actual data being stored within the index in the array. As against, there is a need for more memory in Linked Lists due to storage of additional next and previous referencing elements.
9. In addition memory utilization is inefficient in the array. Conversely, memory utilization is efficient in the linked list.

Algorithm:

```
struct Node
{
    int data;
    struct Node *link;
};
int count(struct node *head)
{
    struct node *temp;
    temp = head;
    int i = 0;
    while(temp != NULL)
    {
        i++;
        temp = temp->next;
    }
    return i;
}
void reverse(struct node *head)
{
```

```
    struct node *temp, *temp2, *prev, *next, *cur;
    int i = 0, j = 0, m = 0;
    temp = temp2 = head; prev = NULL;
    count = count(head);
    while(m < count)
    {
        cur = next = temp;
        for(i = 0; i< k+j && next != NULL; i++)
        {
            next = cur->link;
            cur->link = prev;
            prev = cur;
            cur = next;
            m++;
        }
        j++;
        temp2->link = prev;
        temp2 = temp;
        prev = NULL;
        temp = cur;      //temp = next also will work

    }
}
```

Question No	Question	CO Mapping
Q. NO. 8	What is the difference between a single linked list and a circular linked list representation? Write a function detect the Loop() that checks whether a given double Linked List contains loop and if loop is present then removes the loop and returns true. If the list doesn't contain loop then it returns false. [12] Note: For a loop to be present either the last node's next pointer may keep the address of any other node or the first node's previous pointer may keep the address of any other node in the list or both.	**CO-3**

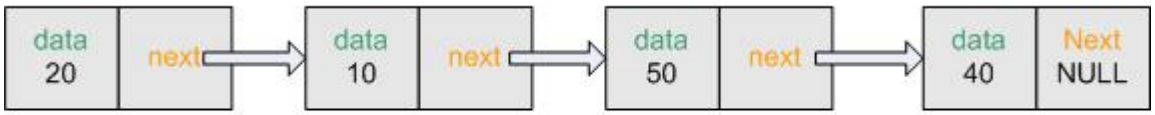
Evaluation Scheme:
Difference: [4 Mark]
Algorithm: [8 Mark]

Solution:

Difference between single link list and circular link list representation:

- **Single Linked list** is a linear data structure which consists of group of nodes in a sequence. Each node has got two parts , a data part- which stores the data and a address part- which stores the address of the next node. Hence forming a chain like

structure. Linked list are used to create trees and graphs.

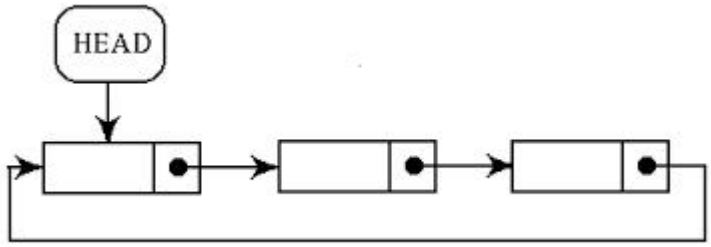


Linked list

- Single way traversal can only be possible. Back traverse is not possible.
- The last node's pointer is always NULL. So, the last node of single linked list is not used very effectively.
- If you delete first node of single linked list then time complexity is constant.

Circular Linked list

- In circular linked list the last node address part holds the address of the first node hence forming a circular chain like structure.



- Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again. you can back traverse in circular linked list which is not possible in linear linked list.
- We can traverse the list from any node i.e. From starting node, middle node and end node too.
- If you delete first node of circular single linked list then time complexity is O(n) because you need to take care of last node's next part.
- The last node's pointer is never NULL but points back to the first node.

Function to detect the Loop() in a given double Linked

```
bool    Loop(struct node *head)
{
    struct node *p, *q;
    p=q=head;
    while (p!= NULL && q!= NULL && q-> next != NULL)
```

```
    {
        p= p-> next;
        q=q->next->next;
        If ( q-> prev ==p-> prev || p == q )
        {
            printf("loop found");
            q->next= NULL;
            return true;
        }
    }
    printf(" No loop found");
    return false;
}
```

Question No	Question	CO Mapping
Q.No:9	**A.** Suppose a computer system has one processor to execute different tasks. Each task has a time of execution. Each task is assigned with a priority number depending upon the type of task: Local Printing (Lowest Priority -1), Web Applications (Priority-2), I/O interfacing (Highest Priority -3). Every time a task is generated, its execution time and priority number are entered and stored. Which data structure can efficiently maintain task waiting for the processor? Write functions for insertion and deletion operations for the tasks with the following conditions. i) A task will be processed first with minimum execution time. ii) A task will be processed first with highest priority. [8]	CO1,CO4

Evaluation Scheme:
Data Structure Name: [1 Mark]
Algorithm for task to be processed with minimum execution time : [3.5 Mark]
Algorithm for task to be processed with highest priority : [3.5 Mark]

Solution:

Priority Queue is the efficient data structure to implement the task scheduling based on their different priority(less execution time or highest priority).

```
Struct node{
int priority;
int data;
int exetime;
```

```
Struct node *next;
} *front=Null;
```

A task will be processed first with minimum execution time.

```
Void insert (int val, int pri, int t1)
{
      Struct node *temp, *q, * prev;
      temp=(Struct node *) malloc (sieof (Struct node));
      temp->priority=pri ;
      temp->value=val;
      temp->exetime=t1
        If    (front==Null)
              {
              temp->next=Null;
              front=temp;
              }
      else
               {
                 q=front
                 while( q->exetime <= t1 && q!=Null)
                             {           prev=q;
                                        q=q->next;
                             }
                  temp->next=q;
                  Prev->next=temp;
                  }
  }
```

```
Struct Node * delete (void)
{
      Struct node *temp:
      If    (front==Null) )
              {
              printf("underflow")
              temp=Null;
              }
      else
        {
      temp=front;
      Front=front->next;
      temp->next=Null;
      }
return(temp);
}
```

A task will be processed first with highest priority.

```
Void insert (int val, int pri, int t1)
```

```
{
    Struct node *temp, *q, * prev;
    temp=(Struct node *) malloc (sieof (Struct node));
    temp->priority=pri ;
    temp->value=val;
    temp->exetime=t1
    If   (front==Null)
         {
         temp->next=Null;
         Front=temp;
         }
    else
          { q=front;
            while( q->priority >=pri && q!=Null)
                      {          prev=q;
                                q=q->next;
                      }
         temp->next=q;
         Prev->next=temp;
         }
}
```

Struct Node * delete (void)

```
{
    Struct node *temp:
         If   (front==Null) )
              {
              printf("underflow")
              temp=Null;
              }
         else
              {
              temp=front;
              Front=front->next;
              temp->next=Null;
              }
    return(temp);
}
```

Q.No:9	**B.** Write insertion and deletion functions to implement an input restricted double ended queue. [4]	CO1,CO4

Evaluation Scheme:
Insertion Function: [2 Marks]
Insertion Function: [2 Marks]

Solution:

```
#define max 100
Int queue[max], front = 0, rear = max-1;
Dequeue Insertion (int element)
{       if (rear == 0)
                {printf("Overflow");
                return; }
          else
            {
             queue[rear] = element;
             rear = rear – 1;
             }
}
int Dequeue DeletionFront( )
{
   if (rear != 0)
         {printf("Underflow");
         exit(0);}
item = queue[front];
front = front + 1;
return item;
}
Int Dequeue DeletionRear( )
{
If (rear == N)
     {   printf("Underflow");
          Exit(0);
   }
item = queue[rear]
rear = rear +1;
return item;    }
```

Question No	**Question**	**CO Mapping**
Q.No:9	A. Write insertion and deletion functions for implementing a priority queue using a two dimensional array. Discuss the time complexities of the functions. [4]	CO1,CO4
Evaluation Scheme: Insertion Function: [2 Marks] Insertion Function: [2 Marks]		

Solution:

```
int pri_que[10][2], rear=-1, front=-1, MAX=10;

void insert_by_priority(int pri_no, int info)
{
      if (rear >= MAX - 1)// Full
      {
            printf("\nQueue overflow no more elements can be inserted");
            return;
      }
      if ((front == -1) && (rear == -1)) // If no element then insert without comparison
      {
            front++;
            rear++;
            pri_que[rear][0] = pri_no;
            Pri_wue[rear][1] = info;
            return;
      }
      else
        {
            int i,j;
            for (i = 0; i <= rear; i++)
            {
                if (pri_no >= pri_que[i][0]) // Comparing existing Priority
                {
                  for (j = rear + 1; j > i; j--)
                  {
                        pri_que[j][0] = pri_que[j - 1][0];
                        pri_que[j][1] = pri_que[j - 1][1];
                  }
                }
                else
                  exit;
             }
             pri_que[i][0] = pri_no;
             pri_que[i][1] = info;
             rear++;
      }
 }

 void delete_by_priority(int pri_no, int info)
 {
       int i;
       if ((front==-1) && (rear==-1)) //No element
       {
            printf("\nQueue is empty no elements to delete");
            return;
       }
```

```
        for (i = 0; i <= rear; i++)
        {
            if (pri_no == pri_que[i][0])
            {
                for (; i < rear; i++)
                {
                    pri_que[i][0] = pri_que[i + 1][0];
                    pri_que[i][1] = pri_que[i + 1][1];
                }
             rear--;
             if (rear == -1)
                front = -1; // Signifies the empty queue after deletion
            return;
            }
        }
        printf("\n%d not found in queue to delete", data);
}
```

Complexity:
Insertion: $O(n^2)$
Deletion: $O(n^2)$
This is because of shifting in two dimensional array.

Q.No:9	B. Write a program to merge two sorted stacks S1 & S2 by using only push and pop functions and without taking any additional data structures. The final merge list to be stored in S1. Both S1 and S2 must be created dynamically. [8]	CO1,CO4

Evaluation Scheme:
Push function: [2 Marks]
Pop function: [2 Marks]
Merge Function: [4 Marks]

Solution:

```
//push1() and pop1() are for push function and pop function for Stack1
//push2() and pop2() are for push function and pop function for stack2

void merge_stacks(struct stack1 * top1, struct stack2 *top2)
{
    int temp,x, count=0,flag=0;
    if(top1->data > top2->data)
        {   temp=pop1();
            push2(temp);
            flag=0;
        }
        else
        {
            temp=pop2();
            push1(temp);
```

```
                flag=1;
            }
        while( top1!= NULL && top2!= NULL)
        {

            if(flag==0)
            {
                temp=pop1();
                while(top2->data> temp)
                {
                    x=pop2();
                    push1(x);
                    count++;
                }
                push2(temp);
                while(count!=0)
                {   x=pop1();
                    push2(x);
                    count--;
                }
            }
            else if( flag==1)
            {   temp=pop2();
                while(top1->data> temp)
                {
                    x=pop1();
                    push2(x);
                    count++;
                }
                push1(temp);
                while(count!=0)
                {   x=pop2();
                    push1(x);
                    count--;
                }
            }
        }
        if(flag==0)
        {   while(top2!=NULL)
            {   x=pop2();
                push1(x);
            }
            display(top1);
        }
        if(flag==1)
        {   display(top1);
        }
}
```

Question No	Question	CO Mapping
Q.No:9	A. Write insertion and deletion functions to implement a QUEUE ADT using STACK ADT.[4]	CO1,CO4

Evaluation Scheme:
Insertion function: [2 Marks]
Deletion function: [2 Marks]

Solution:

```
 struct Queue {
    stack<int> s1, s2;

  void enQueue(int x)
  {
      while (!s1.empty()) {
        s2.push(s1.top());
        s1.pop();
      }

      s1.push(x);

      while (!s2.empty()) {
        s1.push(s2.top());
        s2.pop();
      }
  }


  int deQueue()
  {

      if (s1.empty()) {
        "Q is Empty";
        exit(0);
      }

      int x = s1.top();
      s1.pop();
      return x;
  }
};
```

Q.No:9	B. Write a program to rearrange two sorted stacks S1 & S2 by using only push and pop functions and without taking any additional data structures. [8] Note: **Rearrange**:- Finally, both the arrays will be sorted, but the highest element in S1 should be less than the lowest element in S2.	CO1,CO4

Evaluation Scheme:
Algorithm or pseudo code: [8 Marks]
Solution:

Ans:Consider the input be like-
S1: 13, 18, 20
S2: 5, 15, 25

Final Result should be:
S1: 5,13,**15**
S2: 18,20,**25**

Algorithm:
1.Find out the smallest element among the two stacks.If it is in S2,it has to be pushed inside S1.Let min1 be smallest element of S1 and min2 be smallest element of S2.
S1: 13, 18, 20
S2: 5, 15, 25

min1=S1[0];
min2=S2[0];

The top most element in S1 right now be top1=S1[2].similarly top2=S2[2]
Take two variables to iterate over stack S1(ptr1) and stack S2(ptr2).
A variable elem is taken which is taken to save the value which has to be pushed inside the stack S1 at right position.
S1: 13, 18, 20
S2: 5, **15, 25**

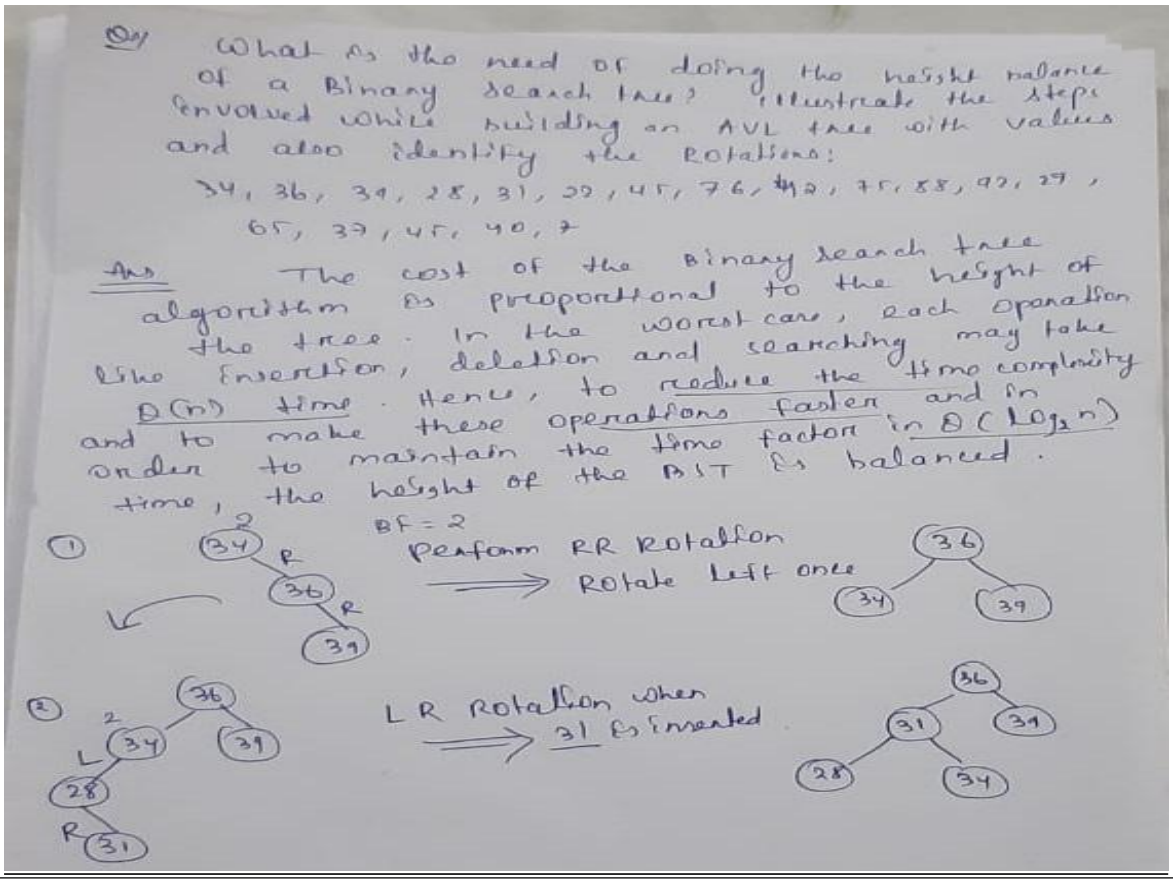
Step2:
int val=pop(S2);
push(S1,val);
S1: 13,18,20,25,15
S2: 5
elem=pop(S2)=5
S1:13,18,20,25,15
S2:
Step3:
val=pop(S1);
push(S2,val);
S1:
S2:15,25,20,18,13
push(S1,elem);
S1:5
S2:15,25,20,18,13
Step4:
Consecutive elements in S2 will be checked and if they are in ascending order, they will be pushed inside stack S1 from S2.
S1:5,13,**18,20,25**
S2:**15**
Same process will follow, next element 15 will occupy space inside S1 after the element

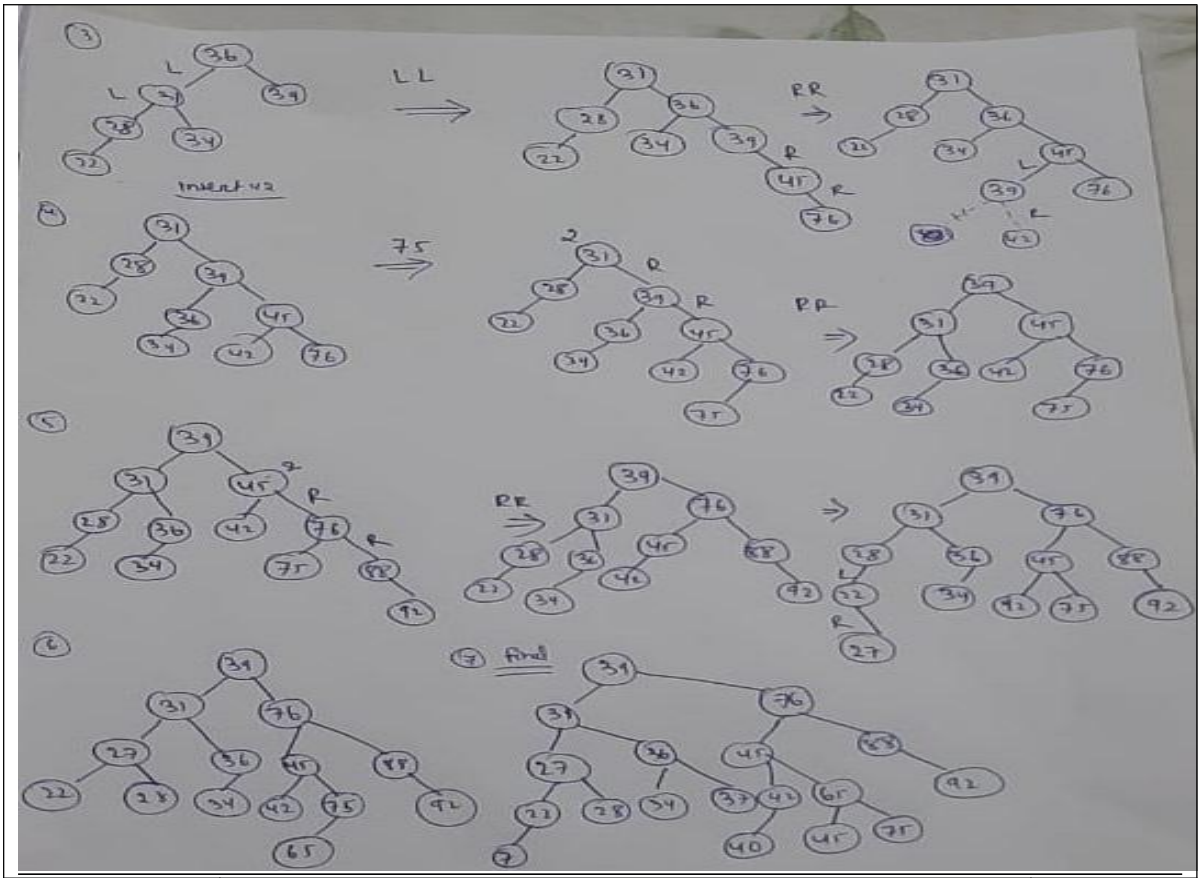
13.
Step5:
elem=pop(S2)=15
val=pop(S1)
push(S2,val)
S1:5,13
S2:18,20,25
push(S1,elem)
S1:5,13,15
S2:18,20,25

Note: Full marks for the C implementation who use above all 5 mentioned steps of the mentioned algorithm. In the program only variable elem is taken, no other data structure is taken as asked in the question.

Question No	**Question**	**CO Mapping**
Q.No 10	A. What is the need of doing height balance of a Binary search tree? Illustrate the steps involved while building an AVL tree with values and also identify the rotations: 34, 36, 39, 28, 31, 22, 45, 76, 42, 75, 88, 92, 27, 65, 37, 45, 40, 7 [7]	CO4,CO6

Evaluation Scheme:
Background of AVL Tree: [1 Mark]
Construction of AVL Tree: [6 Marks]
Solution:





Q.No:10	B. Write a function to construct a fully complete binary tree from the given level order traversal sequence of the tree. [5]	CO4,CO6

Evaluation Scheme:
Algorithm/ pseudo code: [5 Mark]

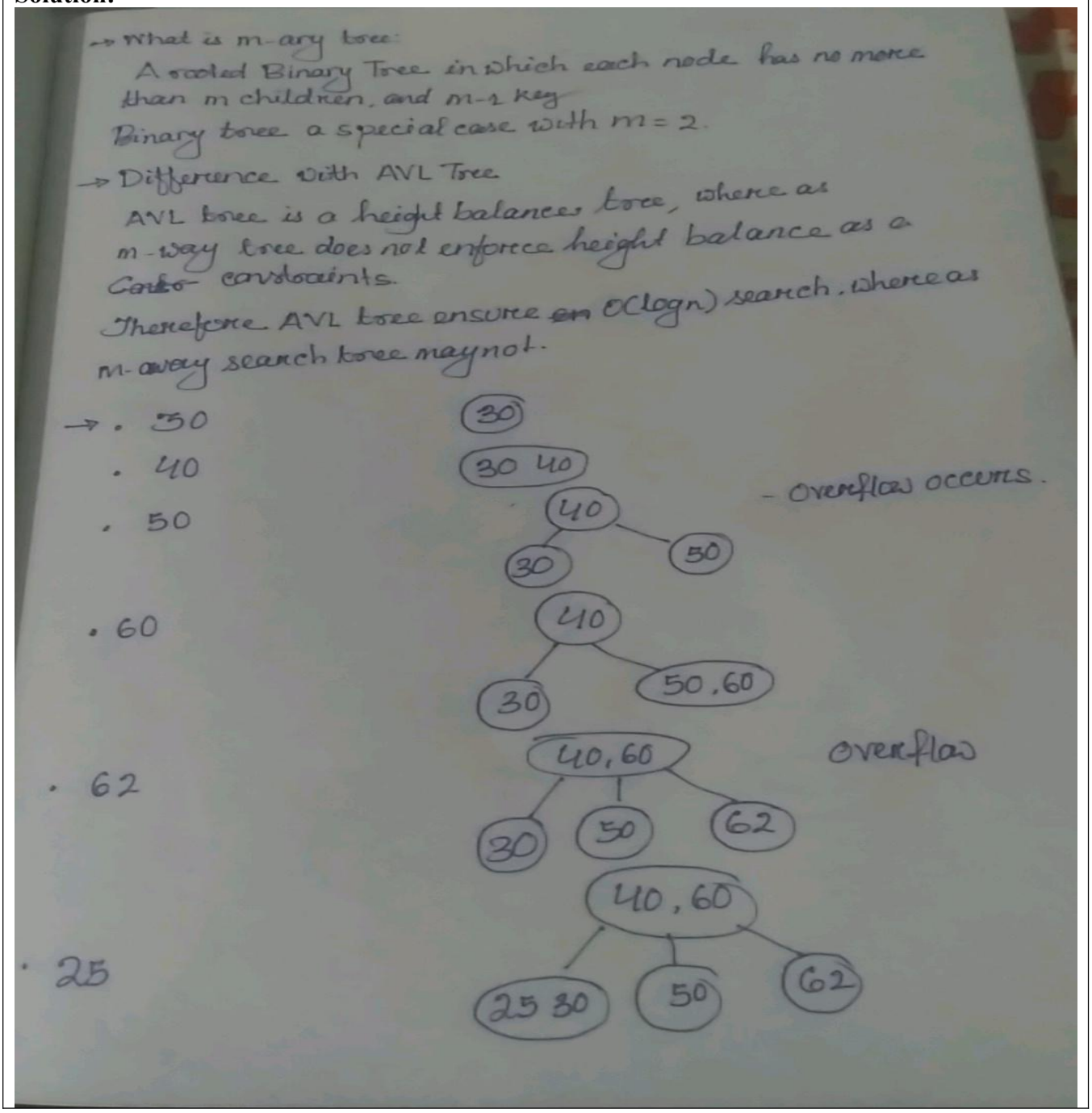
Solution:

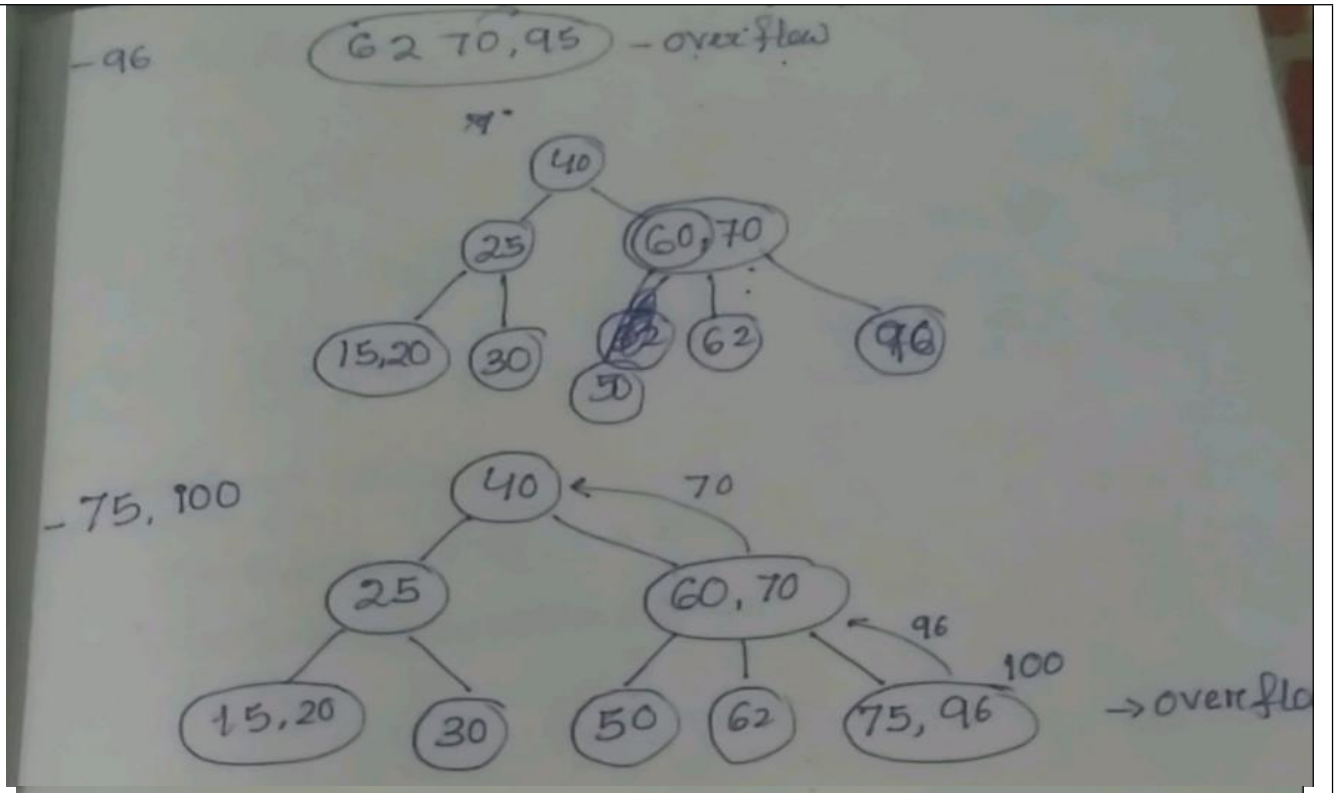
```
Node* newNode(int data)
{
   Node* node = (Node*)malloc(sizeof(Node));
   node->data = data;
   node->left = node->right = NULL;
   return (node);
}
// Function to insert nodes in level order
Node* insertLevelOrder(int arr[], Node* root, int i, int n)
{
   if (i < n)      // Base case for recursion
   {
      Node* temp = newNode(arr[i]);
      root = temp;
      root->left = insertLevelOrder(arr,   root->left, 2 * i + 1, n);    // insert left child
      root->right = insertLevelOrder(arr, root->right, 2 * i + 2, n);   // insert right child
   }
   return root;
}
```

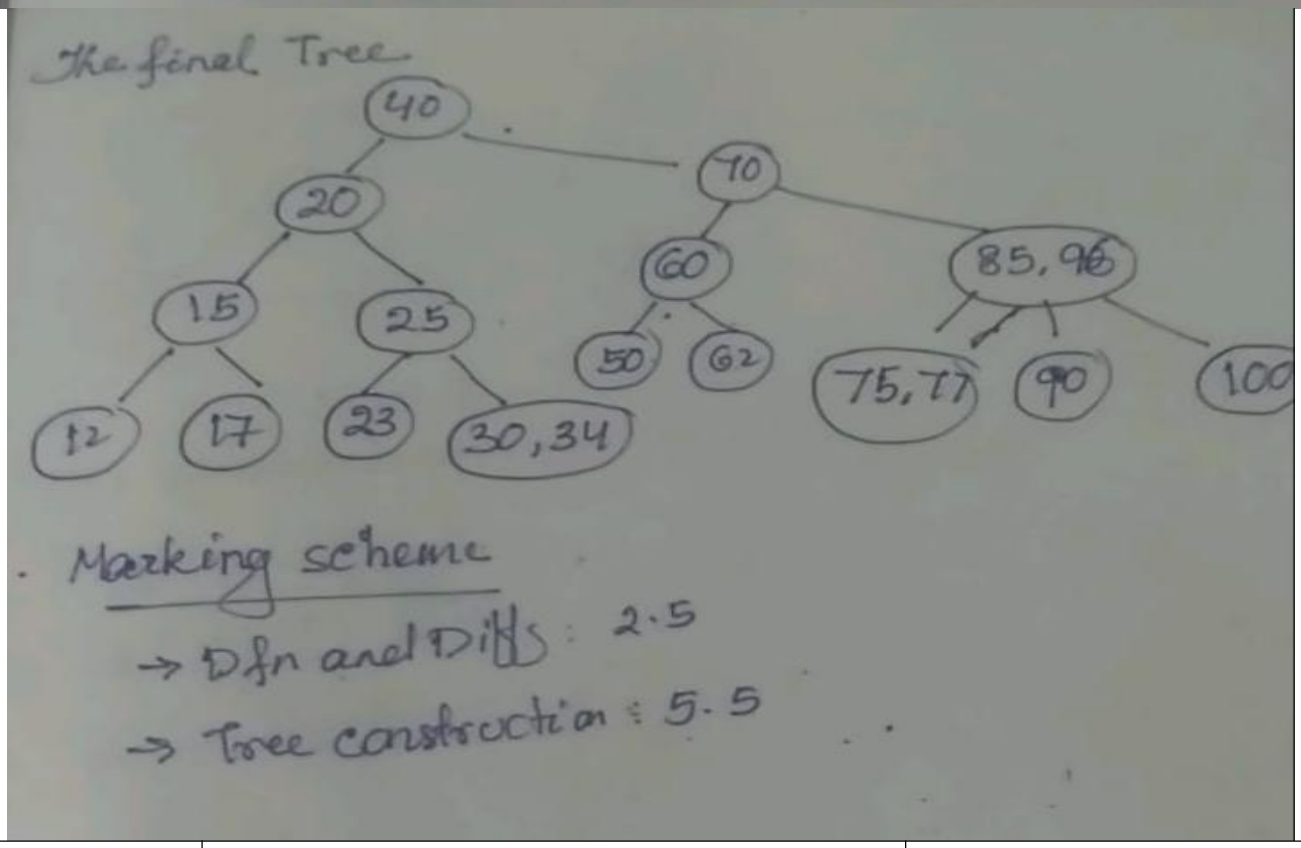
Question No	Question	CO Mapping
Q.No: 10	A. What is an m-way search tree? How is it different from AVL tree? Create a B-tree of order 3 with the following elements. [8] 30,40,50,60,62,25,20,15,70,96,75, 100, 85, 62, 34, 12, 23, 90, 17, 15, 77	CO4,CO6

Evaluation Scheme:
Definition of m-way search tree: [1 Mark]
Difference of m-way search tree and AVL Tree: [1 Mark]
Construction of B Tree: [6 Marks]

Solution:







Q.No: 10	B. Write a function to check a given tree is a binary search tree or not.[4]	CO4,CO6

Evaluation Scheme:
Algorithm/ pseudo code: [4 Marks]

Solution:

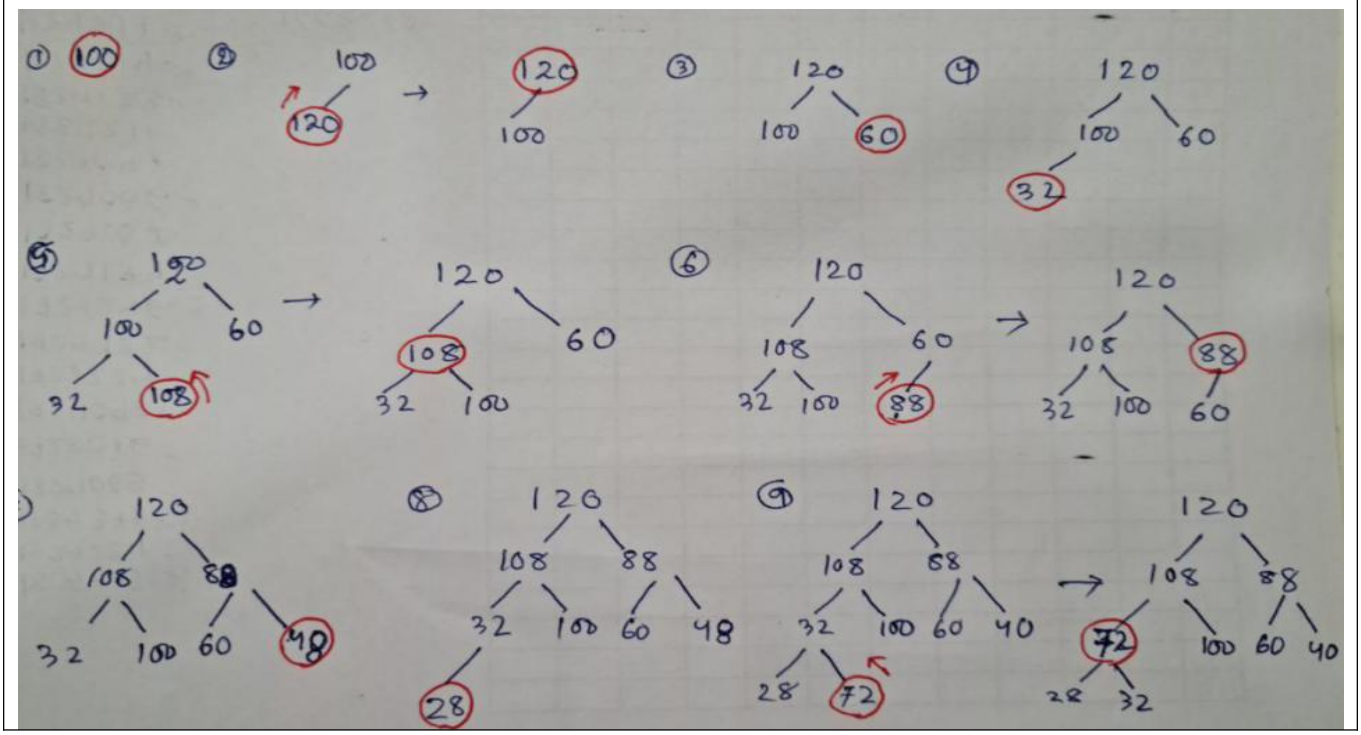
```
bool isBST(Node* node, int minKey, int maxKey)
{   if (node == NULL)      // base case
        return true;
    if (node->data < minKey || node->data > maxKey)       // if node's value fall outside valid range
        return false;
 // recursively check left and right subtrees with updated range
    return isBST(node->left, minKey, node->data) && isBST(node->right, node->data, maxKey);
}
```

Question No	Question	CO Mapping
Q.No:10	A. What is a max heap tree and min heap tree. Construct a max heap with the following data elements. [4] 100, 120, 60, 32, 108, 88, 48, 28, 72.	CO4,CO6

Evaluation Scheme:
Definition: [1 Mark]
Heap Construction: [3 Marks]

Solution:
It's a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.

Min-Heap − Where the value of the root node is less than or equal to either of its children.
Max-Heap − Where the value of the root node is greater than or equal to either of its children.
Both **trees** are constructed using the same input and order of arrival.



Q.No:10	B. Write a program to convert a given postfix arithmetic expression into its equivalent prefix arithmetic expression using expression tree. [8]	CO4,CO6

Note: Program to convert postfix expression to its equivalent prefix expression is not not included in the 3rd semester DSA syllabus.
Evaluation Scheme: Sentence type solution or example type solution can be accepted as the answer to this question. Lenient marking can be done for this question.

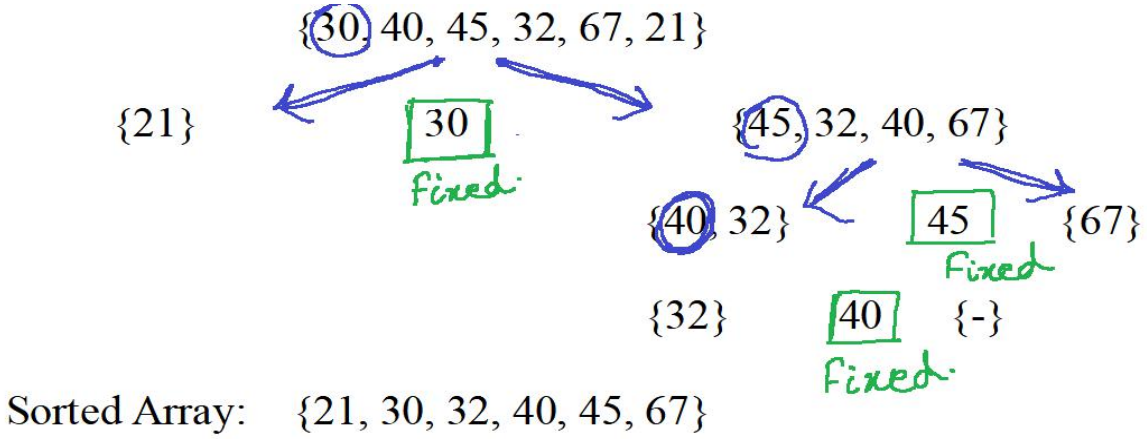
Question No	Question	CO Mapping
Q.No:11	A. Write the pseudo code for Quick sort including partition() function. Also illustrate the step by-step process of partition() function with a given array: {30, 40, 45, 32, 67, 21}, considering first element as the pivot element. [6]	CO4, CO5,CO6

Evaluation Scheme:
Quick Sort(): [Marks 3]
Solving Problem Instance: [Marks 3]

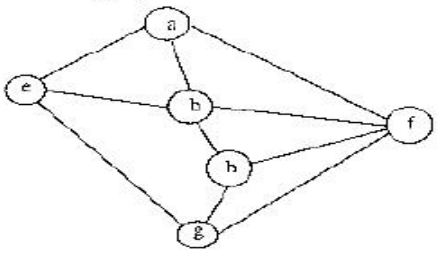
Solution:

```
quickSort(arr[], low, high){
      if (low < high){
            q = partition(arr, low, high);
            quickSort(arr, low, q - 1);
            quickSort(arr, q + 1, high);
      }
}
int partition(int a[], int low, int high){
    int i=low, j=high, pivot=a[low], temp;
    while(true){
            while(a[i]<=pivot)
                i++;
            while(a[j]>=pivot)
                j--;
            if(i<j){
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
            else
            {
                a[low]= a[j];
                a[j]=pivot;
                return(j);
            }
    }
}
```

Illustration of quick sort in a step by-step process {30, 40, 45, 32, 67, 21}, considering first element as the pivot element.



Sorted Array: {21, 30, 32, 40, 45, 67}

Q.No:11	B. Write the DFS traversal algorithm. Show all the steps to find DFS traversal of the given graph. The traversal starts from vertex h. [3] 	CO4, CO5,CO6

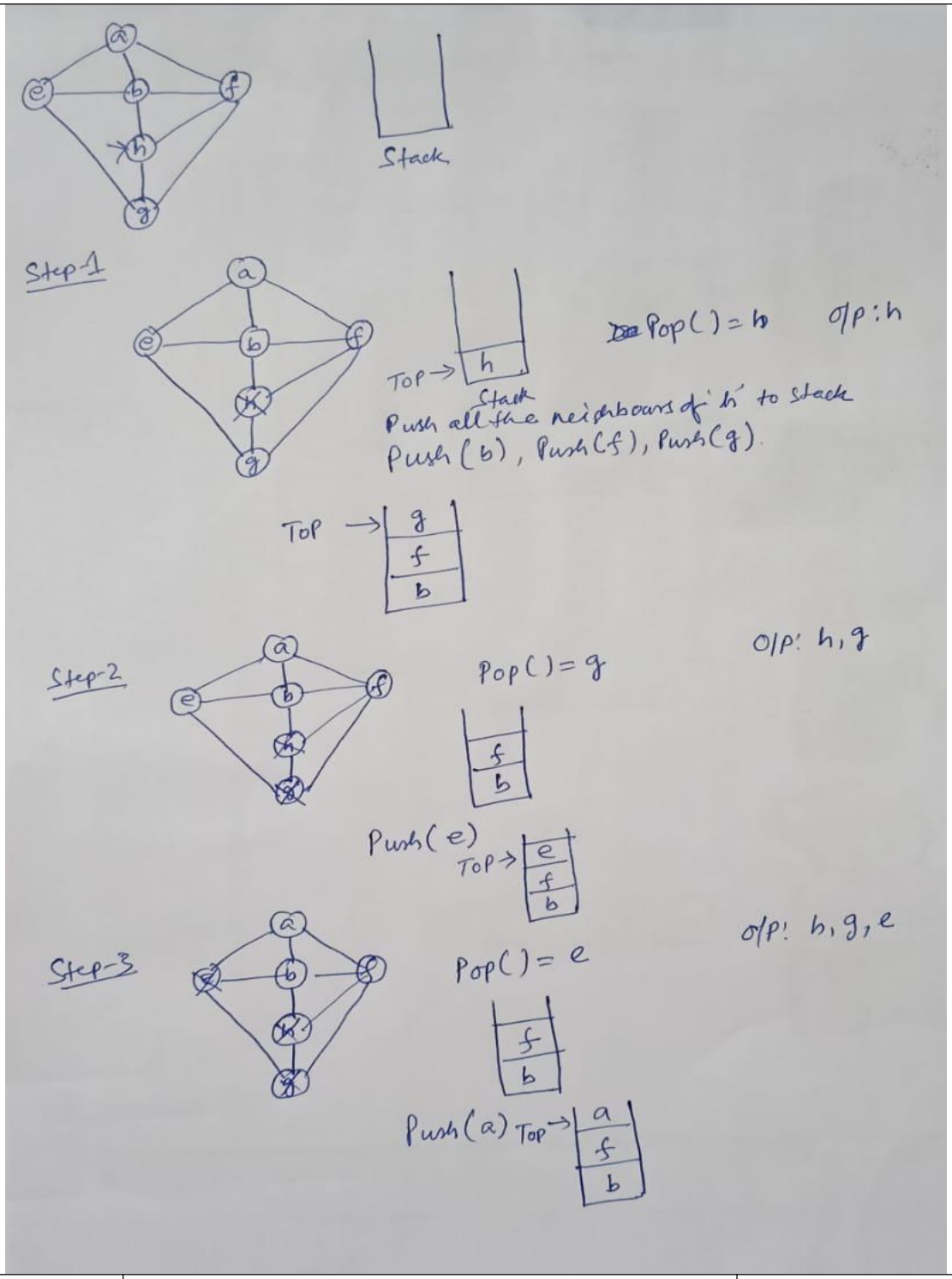
Evaluation Scheme:
Algorithm: [Marks 1.5]
Solving Problem Instance: [Marks 1.5]

Solution:

This algorithm executes a depth-first search on a graph G beginning at a starting node A.

1. Initialize all nodes to the ready state (STATUS = 1).
2. Push the starting node A onto STACK and change its status to the waiting state (STATUS = 2).
3. Repeat Steps 4 and 5 until STACK is empty.
4. Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5. Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).

   [End of Step 3 loop.]
6. Exit.



Q.No:11	C. Write a function to implement chaining(collision resolution technique) in hashing. [3]	CO4, CO5,CO6

Note: Implementation of hashing is not not included in the 3rd semester DSA syllabus.
Evaluation Scheme: Sentence type solution or example type solution can be accepted as the answer to this question. Lenient marking can be done for this question.

Solution:

```
int slotCount ;
```

```
//node definition
struct node
{
     int key;
     struct node *next;
};

//hash definition
struct hash
{
   struct node *head;
   int count;
} *hashTable ;

struct node * createNode(int k) //New node creation
{
   struct node *newnode;
   newnode = (struct node *)malloc(sizeof(struct node));
   newnode->key = k;
   newnode->next = NULL;
   return newnode;
}

void insertToHash(int k)
{
   int hashIndex = k % slotCount;
   struct node *newnode =    createNode(k);
   /* head of list for the bucket with index "hashIndex" */
   if (!hashTable[hashIndex].head)
   {
      hashTable[hashIndex].head = newnode;
      hashTable[hashIndex].count = 1;
      return;
   }
   /* adding new node to the list */
   newnode->next = (hashTable[hashIndex].head);

   /* update the head of the list and no of    nodes in the current bucket */
   hashTable[hashIndex].head = newnode;
   hashTable[hashIndex].count++;
}

void searchInHash(int k)
{
   int hashIndex = k % slotCount, flag = 0;
   struct node *myNode;
   myNode = hashTable[hashIndex].head;
   if (!myNode)
   {
        printf("Search element unavailable in hash table\n");
```

```
        return;
    }
    while (myNode != NULL)
    {
     if (myNode->info == k)
     {
        printf("Element is    : %d\n", myNode->info);
        flag = 1;
        break;
      }
      myNode = myNode->next;
     }
     if (!flag)
        printf("Search element unavailable in hash table\n");
}
```

Question No	Question	CO Mapping
Q.No:11	A. Write the pseudo code for Merge sort including merging () function. Also illustrate the step-by-step process of merge sort with a given array: {30, 40, 45, 32, 67, 21, 89, 24, 100}. Discuss the time complexity of insertion sort, quick sort, and merge sort. [6]	CO4, CO5,CO6

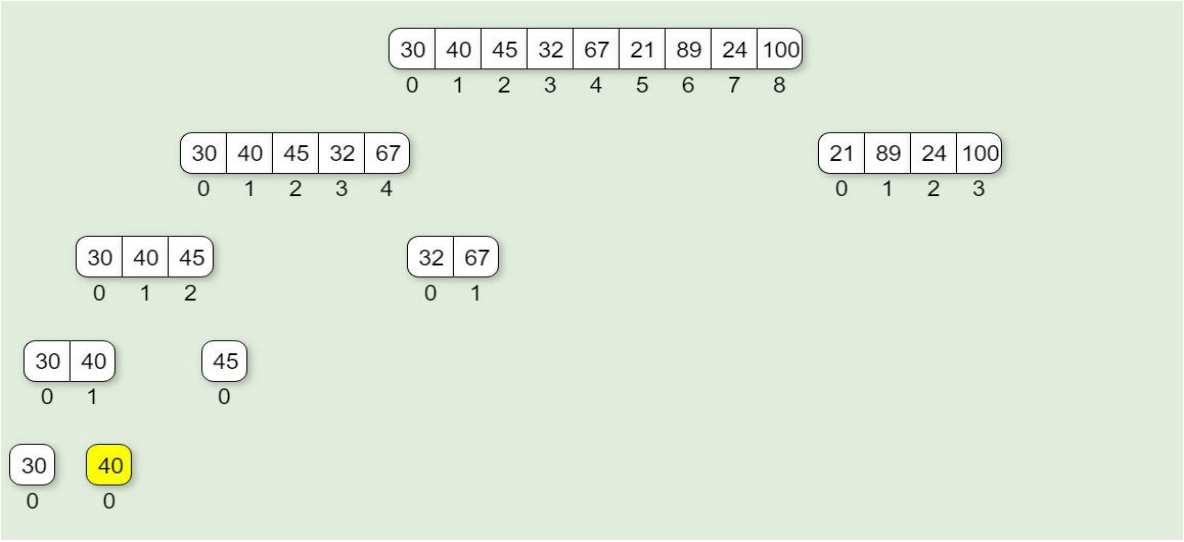
Evaluation Scheme:
Merge Sort(): [Marks 3]
Solving Problem Instance: [Marks 3]

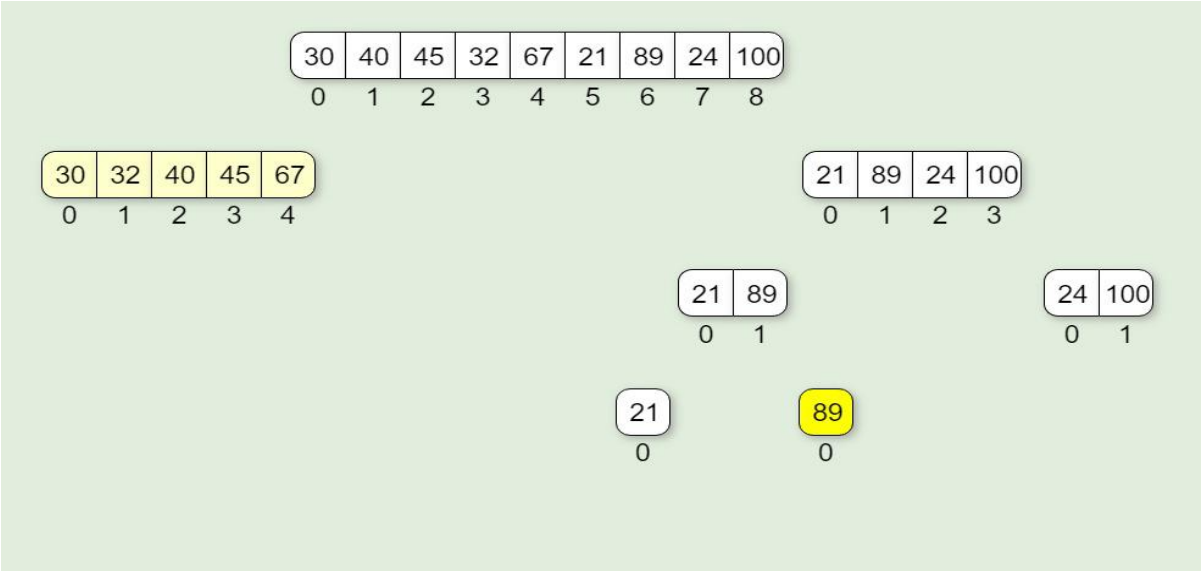
Solution:

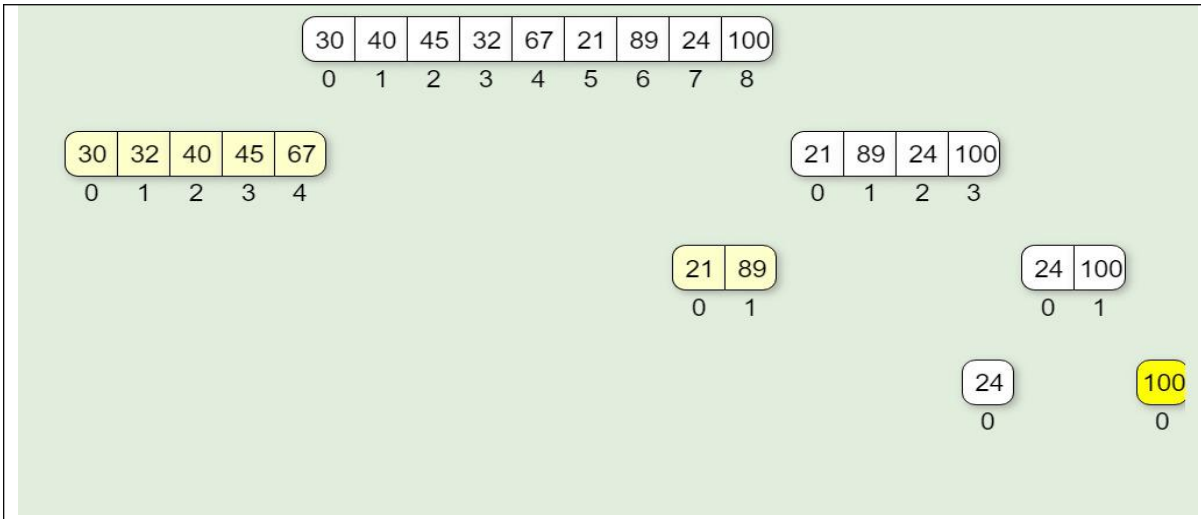
```
void merging(int * a, int low, int mid, int high) {
    int l1, l2, i;

    for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
        if(a[l1] <= a[l2])
            b[i] = a[l1++];
        else
            b[i] = a[l2++];
    }

    while(l1 <= mid)
        b[i++] = a[l1++];

    while(l2 <= high)
        b[i++] = a[l2++];

    for(i = low; i <= high; i++)
        a[i] = b[i];
```

```
}

void merge_sort(int low, int high) {
    int mid;

    if(low < high) {
        mid = (low + high) / 2;
        merge_sort(low, mid);
        merge_sort(mid+1, high);
        merging(low, mid, high);
    } else {
        return;
    }
}
```

Illustration of step-by-step process of merge sort with a given array: {30, 40, 45, 32, 67, 21, 89, 24, 100}







Comparison:

Sorting Name	Best Case	Average Case	Worst Case
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Quick Sort	$\Omega(n\log n)$	$\Theta(n\log_e n)$	$O(n^2)$
Merge Sort	$\Omega(n\log_2 n)$	$\Theta(n\log_2 n)$	$O(n\log_2 n)$

Q.No:11	B. Write a function to implement linear probing (collision resolution technique) in hashing. [3]	CO4, CO5,CO6

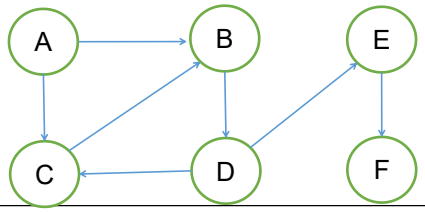
Note: Implementation of hashing is not not included in the 3rd semester DSA syllabus.
Evaluation Scheme: Sentence type solution or example type solution can be accepted as the answer to this question. Lenient marking can be done for this question.

Solution:

```
#define M 15 //any number
struct DataItem
{
    INT Key; //Unique
    INT Value;
} HT[M];

VOID INITIALIZE()
BEGIN
  FOR EACH DataItem DI in HT
     DI.Key = Ø
  END FOR
END

INT LOADFACTOR()
BEGIN
  C <- 0
  FOR EACH DataItem DI in HT
     IF (DI.Key != Ø ) THEN
```

```
            C = C+1
        END IF
    END FOR
    RETURN FLOOR(C/M)
END

VOID INSERT(DataItem DI)
BEGIN
     IF (LOADFACTOR() == 1) THEN
           DISPLAY “Hash Table Full”
           EXIT()
     END IF

    // NO COLLISION
     SET H <- DI.Key    MOD M
     IF (HT[H].Key == Ø) THEN
          HT[H] = DI
          EXIT()
     END IF

     //COLLISION OCCOURED - FORWARD
    I <- H +1 //Next hash address
    WHILE (I < M) DO
       K <- I MOD M
       IF (HT[K].Key == Ø) THEN
          HT[K] = DI
          EXIT()
       END IF
       I = I+1
    END WHILE

    //COLLISION OCCOURED - REVERSE
    I <- H - 1
    WHILE (I >=0) DO
     K <- I MOD M
       IF (HT[K].Key == Ø) THEN
          HT[K] = DI
          EXIT()
       END IF
     I = I - 1
    END WHILE
END

BOOL SEARCH(int k)
BEGIN
     IF (LOADFACTOR() == 0) THEN
          DISPLAY “Hash Table Empty”
           EXIT()
     END IF
```

```
    // NO COLLISION
     SET H <- k    MOD M
     IF (HT[H].Key == k) THEN
          DISPLAY "Success"
          RETURN TRUE
     END IF

     //COLLISION OCCOURED
     I <- H +1 //Next hash address
     WHILE (I <M) DO
       K <- I MOD M
       IF (HT[K].Key == k) THEN
          DISPLAY "Success"
          RETURN TRUE
       END IF
       I = I+1
     END WHILE

     //COLLISION OCCOURED
     I <- H - 1
     WHILE (I >=0) DO
        K <- I MOD M
        IF (HT[K].Key == k) THEN
          DISPLAY "Success"
          RETURN TRUE
        END IF
     I = I - 1
   END WHILE
   DISPLAY "No Success"
  RETURN FALSE
END
```

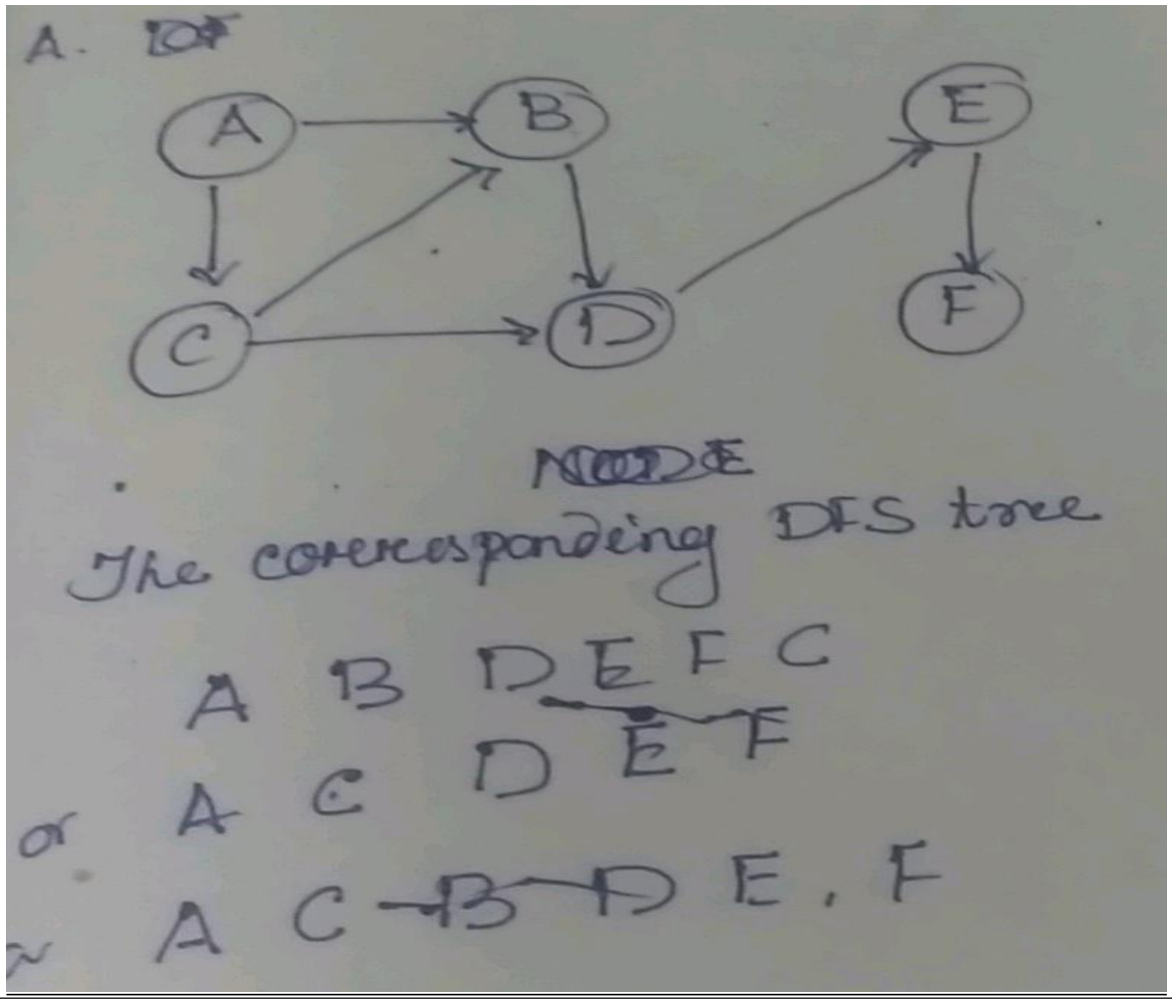
Q.No:11	C. Write the Depth First Search (DFS) graph traversal algorithm/ pseudo code. Discuss the steps of DFS algorithm with the following graph, where A is considered as starting vertex. [3] 	CO4, CO5,CO6

Evaluation Scheme:
Algorithm: [Marks 1.5]
Solving Problem Instance: [Marks 1.5]

Solution:

```
procedure DFS_iterative(G, v) is
      let S be a stack
      S.push(v)
```

```
while S is not empty do
    v = S.pop()
    if v is not labeled as discovered then
        label v as discovered
        for all edges from v to w in G.adjacentEdges(v) do
            S.push(w)
```



Question No	Question	CO Mapping
Q.No:11	A. What is Hashing? Explain the collision with example. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 8 values into an empty hash table, the table is as shown below.	CO_4, CO_5,CO_6

0	
1	
2	12
3	13
4	2
5	3

	6 | 23 7 | 5 8 | 18 9 | 15 Find out the possible order in which the key values could have been inserted in the table (justify the answer with proper explanation). Write function to implement linear probing. [8]	

Note: Implementation of hashing is not not included in the 3rd semester DSA syllabus.

Evaluation Scheme:

Definition and Problem Solving: [4 Marks]

Algorithm or pseudo code: [4 Marks]

Note: Sentence type solution or example type solution can be accepted as the answer to this question. Lenient marking can be done for this question.

Solution:

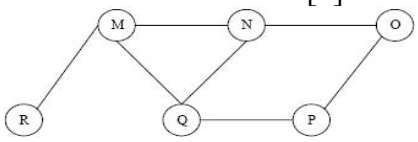
Hashing: It is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given key value to an address for faster access of elements.

Collision: When a hash function generates more than one address for a particular key, collisions occur. To resolve this, the next available empty slot in the hash table is assigned to the current hash value. The most common methods are open addressing, chaining, probabilistic hashing, perfect hashing and coalesced hashing technique.

```
/* to insert an element in the hash table */
void insert(int key, int value)
    {
        int index = hashcode(key);
        int i = index;

        /* creating new item to insert in the hash table array */
        struct item *new_item = (struct item*) malloc(sizeof(struct item));
        new_item->key = key;
        new_item->value = value;

        /* probing through the array until we reach an empty space */
        while (array[i].flag == 1)
        {
      if (array[i].data->key == key)
            {

        /* case where already existing key matches the given key */
        printf("\n Key already exists, hence updating its value \n");
        array[i].data->value = value;
        return;

      }
```

```
        i = (i + 1) % max;
        if (i == index)
                    {
            printf("\n Hash table is full, cannot insert any more item \n");
            return;
        }
       }

            array[i].flag = 1;
            array[i].data = new_item;
            size++;
            printf("\n Key (%d) has been inserted \n", key);
       }

       int hashcode(int key)
   {
        return (key % max);
   }
```

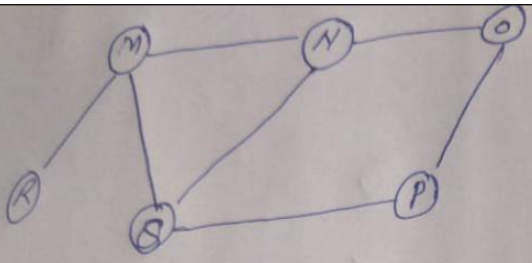
The order of insertion into the hash table is: 12, 13, 2, 3, 23, 5, 18, and 15. Collision occurs at 2, 3, 23, 5, and 15.

Q.No:11	B. Write the BFS traversal algorithm. Show all the steps to find BFS traversal of the given graph. The traversal starts from R. [4] 	CO4, CO5,CO6

Evaluation Scheme:
Algorithm: [Marks 2]
Solving Problem Instance: [Marks 2]

Solution:

```
BFS (G, s)              //Where G is the graph and s is the source node
        let Q be queue.
        Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.

        mark s as visited.
        while ( Q is not empty)
              //Removing that vertex from queue,whose neighbour will be visited now
              v   =   Q.dequeue( )

             //processing all the neighbours of v
             for all neighbours w of v in Graph G
                   if w is not visited
                                Q.enqueue( w )        //Stores w in Q to further visit its neighbour
                                mark w as visited.
```



BFS: (i) Start Node R.

Visited Node: R, M, S, N, O, P

QUEUE: R

(ii) QUEUE : M S, N

(iii) QUEE : N, O

(iv) QUE : O, P

(v) QUE : P = Empty