Code -

G. S. Mandal's

Maharashtra Institute of Technology, Aurangabad

(An Autonomous Institute)

END SEMESTER EXAMINATION

Second Year B.Tech (AIDS) – Feb/Mar-2023

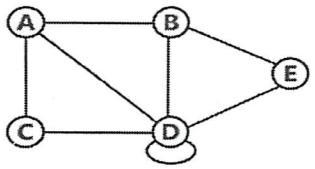
Course Code :AID201 **Course Name : Data Structure and Algorithms**
Duration : 2 Hrs **Max. Marks : 50** **Date :**
Instructions :

i) All questions are compulsory
ii) Assume suitable data wherever necessary and clearly state it
iii) Figures to right indicate full marks

Q. 1	Answer any five(Marks:10)	Stepwise Marking Scheme

a)	Define structure data type with syntax.
	Answer :
	Structure: A structure is a keyword that creates user-defined data types in C. A structure creates a data type that can be used to group items of possibly different types into a single type. We can use this data type to store dates of different attributes of different data types.
	Syntax:
	struct Structure_name
	{
	Datatype member1;
	Datatype member2;
	Datatype member3;
	};
		1
		1

b)	Define following terminologies. Complete binary tree, Binary search tree. **Answer:** **Complete binary tree :** A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left. A complete binary tree is just like a full binary tree, but with two major differences 1. All the leaf elements must lean towards the left.	1

	2. The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree. Binary search tree: In computer science, a binary search tree (BST), also called an ordered or sorted binary tree, is a rooted binary tree data structure with the key of each internal node being greater than all the keys in the respective node's left subtree and less than the ones in its right subtree.	1
c)	Define algorithm and complexity of algorithm. **Answer:** **Algorithm:** An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines. **Complexity of algorithm:** The term algorithm complexity measures how many steps are required by the algorithm to solve the given problem. It evaluates the order of count of operations executed by an algorithm as a function of input data size. O(f) notation represents the complexity of an algorithm, which is also termed as an Asymptotic notation or "**Big O**" notation.	1 1
d)	The following sequence of operation is performed on stack : push(1), push(2), pop, push(1), push(2), pop, pop, pop, push(2), pop. The sequence of popped out values are? **Answer:** As per stack push and pop operation following is the sequence of values popped out from stack: 2 2 1 1 2	2
e)	Represent the following graphs using adjacency matrix.  **Answer:** The graphs is represent using adjacency matrix as below. $\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$	2

f)	What are the advantages of linked list? **Answer:** a. Save memory space and easy to maintain b. It is possible to retrieve the element at a particular index c. It is possible to traverse the list in the order of increasing index. C203.1 BTL 1 d. It is possible to change the element at a particular index to a different value, without affecting any other elements.	2
g)	Write best case, average case and worst case complexity of merge sort and quick sort. **Answer:** Algorithm — Best Case — Average case — Worst case Merge sort — O(n log n) — O(n log n) — O(n log n) Quick sort — O(n log n) — O(n log n) — $O(n^2)$	1 1
h)	Write short note on hash table. **Answer:** Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data. Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.	2
Q.2	**Solve any two.**	
a)	What are disadvantages of static memory location? Explain realloc and free function with example. **Answer:** **Disadvantages of static memory location:** 1. In Static memory, the memory size cannot be changed. So if size is 10 memory location and we insert 4 elements then remaining memory is waste. 2. During runtime if memory requirement need to be change then it is not possible in static memory allocation. **1. realloc ( )** The "realloc" method in C is used to dynamically change the memory allocation	1 1

	of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory. re-allocation of memory maintains the already present value and new blocks will be initialized with the default garbage value. Syntax: **ptr = realloc(ptr, newSize);** Example: **ptr = realloc(ptr, n * sizeof(int));** **2. free( )** The **"free"** method in C is used to dynamically **de-allocate** the memory. It helps to reduce wastage of memory by freeing it. Syntax: **free(ptr);**	1 1
b)	Convert following infix expression to postfix expression using stack: **(A + B / C * (D + E) – F)** **Answer:**	

Expression	Stack	Prefix	
(	(		
A	(	A	
+	(+	A	1
B	(+	AB	
/	(+/	AB	
C	(+/	ABC	
*	(+*	ABC/	1
(	(+*(	ABC/	
D	(+*(	ABC/D	
+	(+*(+	ABC/D	
E	(+*(+	ABC/DE	1
)	(+*	ABC/DE+	
-	(-	ABC/DE+*+	
F	(-	ABC/DE+*+F	
)		ABC/DE+*+F-	1

c)	Evaluate following postfix expression using stack: **5 2 $ 6 2 / * 4 5 * + 5 +** **Answer:**	

Symbol	Operand1	Operand2	Value	Stack	
5				5	
2				5,2	
$	5	2	25	25	1
6				25,6	

		2				25,6,2	1
		/	6	2	3	25,3	
		*	25	3	75	75	
		4				75,4	
		5				75,4,5	1
		*	4	5	20	75,20	
		+	75	20	95	95	
		5				95,5	
		+	95	5	100	100	1
Q.3	**Solve any one.**						

a)	Define queue and write implementation code of enqueue, dequeue and display function using single linked list in C language.
	Answer
	Queue : Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first. We insert data at rear end and delete the element at front end.
	`struct node {`
	`  int data;`
	`  struct node * next;`
	`};`
	
	`struct node * front = NULL;`
	`struct node * rear = NULL;`
	
	`void enqueue(int value) {`
	`  struct node * ptr;`
	`  ptr = (struct node * ) malloc(sizeof(struct node));`
	`  ptr - > data = value;`
	`  ptr - > next = NULL;`
	`  if ((front == NULL) && (rear == NULL)) {`
	`  front = rear = ptr;`
	`  } else {`
	`  rear - > next = ptr;`
	`  rear = ptr;`
	`  }`
	`  printf("Node is Inserted\n\n");`
	`}`
	
	`int dequeue() {`
	`  if (front == NULL) {`
	`  printf("\nUnderflow\n");`
	`  return -1;`
	`  } else {`
	`  struct node * temp = front;`
	`  int temp_data = front - > data;`
	`  front = front - > next;`
								2
								2
								2

```
        free(temp);
        return temp_data;
    }
}

void display() {
    struct node * temp;
    if ((front == NULL) && (rear == NULL)) {
        printf("\nQueue is Empty\n");
    } else {
        printf("The queue is \n");
        temp = front;
        while (temp) {
            printf("%d--->", temp - > data);
            temp = temp - > next;
        }
        printf("NULL\n\n");
    }
}
```

2

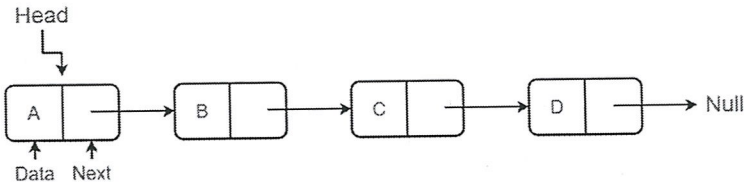
b) What is linked list? Write following functions in C to implement doubly linked list.

1) Insert node at specific position
2) Insert node at end of list
3) Delete node at beginning of list

Answer:

A Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers. They include a series of connected nodes. Here, each node stores the data and the address of the next node.

2



Insert node at specific position:

```
void insert_specified(int item)
{

  struct node *ptr = (struct node *)malloc(sizeof(struct node));
  struct node *temp;
  int i, loc;
    printf("\nEnter the location\n");
     scanf("%d",&loc);
     temp=head;
     for(i=0;i<loc;i++)
     {
        temp = temp->next;
        if(temp == NULL)
```

2

```
            {
                printf("\ncan't insert\n");
                return;
            }
        }
        ptr->data = item;
        ptr->next = temp->next;
        ptr -> prev = temp;
        temp->next = ptr;
        temp->next->prev=ptr;
        printf("Node Inserted\n");
    }
```

Insert node at end of list : 2

```
void insertlast(int item)
{

  struct node *ptr = (struct node *) malloc(sizeof(struct node));
  struct node *temp;
  ptr->data=item;
    if(head == NULL)
    {
      ptr->next = NULL;
      ptr->prev = NULL;
      head = ptr;
    }
    else
    {
      temp = head;
      while(temp->next!=NULL)
      {
        temp = temp->next;
      }
      temp->next = ptr;
      ptr ->prev=temp;
      ptr->next = NULL;
    }
printf("\nNode Inserted\n");
}
```

Delete node at beginning of list: 2

```
void beginning_delete()
{
   struct node *ptr;
 if(head->next == NULL)
   {
     head = NULL;
     free(head);
     printf("\nNode Deleted\n");
   }
   else
```

```
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

Q.4 **Solve any two.**

a) Suppose the following list of letters is inserted in order into an empty binary search tree

25,15,10,22,50,35,70,31,44,90,66,12,4,18,24

(i) Draw the final BST Tree.

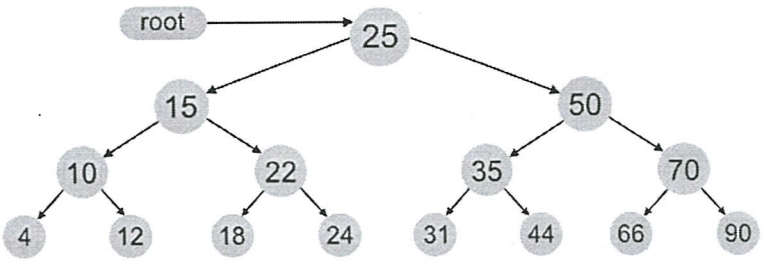
(ii) Find the preorder, inorder and post order traversal of above BST

Answer:

InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90 — 1

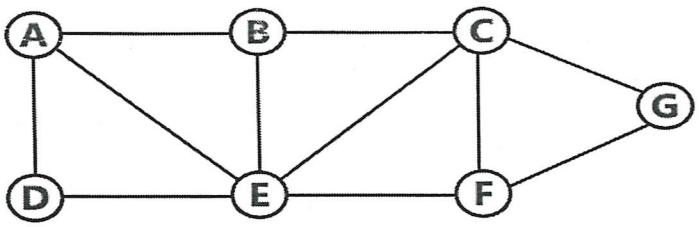
A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90 — 1

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25 — 1



1

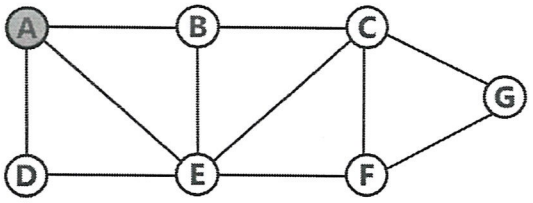
b) Consider following graph perform DFS traversal. Show Step by step solution

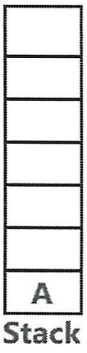


Answer: The step by step solution of given graph using DFS is given as:

Step 1:

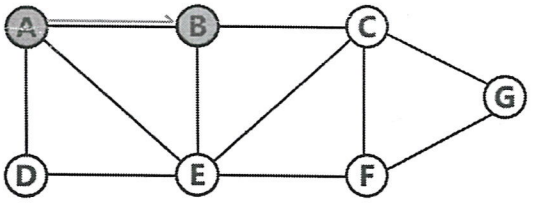
- Select the vertex **A** as starting point (visit **A**).
- Push **A** on to the Stack.

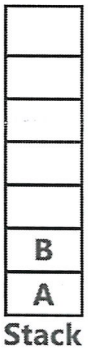




Step 2:

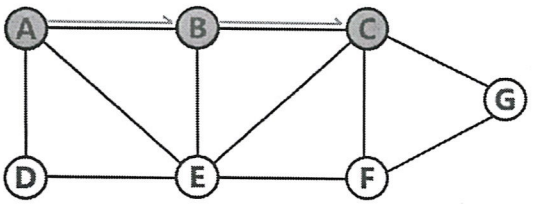
- Visit any adjacent vertex of **A** which is not visited (**B**).
- Push newly visited vertex B on to the Stack.

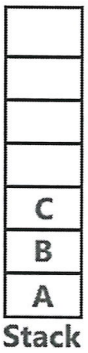




Step 3:

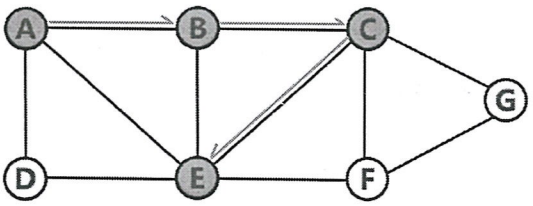
- Visit any adjacent vertext of **B** which is not visited (**C**).
- Push C on to the Stack.

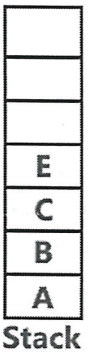




Step 4:

- Visit any adjacent vertext of **C** which is not visited (**E**).
- Push E on to the Stack

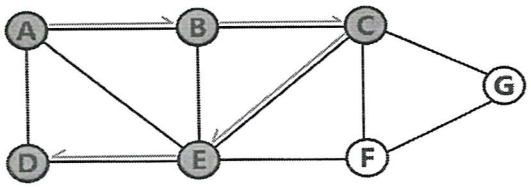


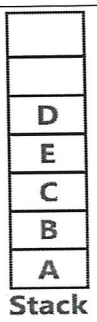


1

Step 5:

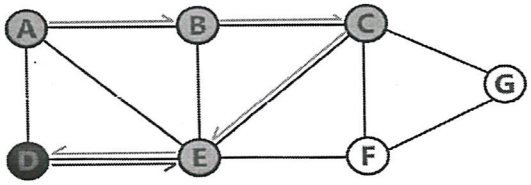
- Visit any adjacent vertext of **E** which is not visited (**D**).
- Push D on to the Stack

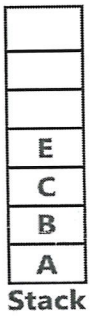




Step 6:

- There is no new vertiex to be visited from D. So use back track.
- Pop D from the Stack.

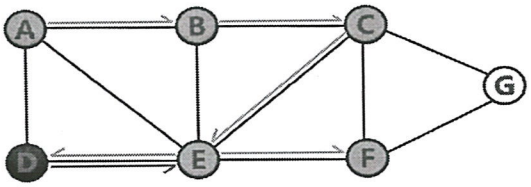


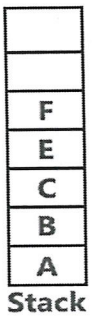


1

Step 7:

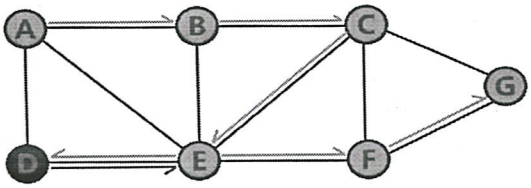
- Visit any adjacent vertex of **E** which is not visited (**F**).
- Push **F** on to the Stack.

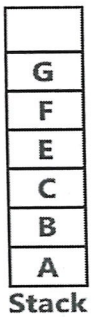




Step 8:

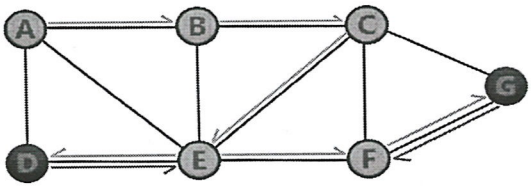
- Visit any adjacent vertex of **F** which is not visited (**G**).
- Push **G** on to the Stack.

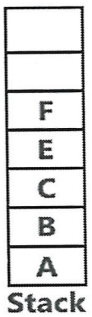




Step 9:

- There is no new vertiex to be visited from G. So use back track.
- Pop G from the Stack.

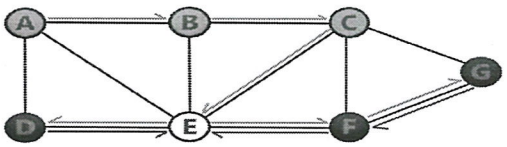


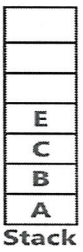


1

Step 10:

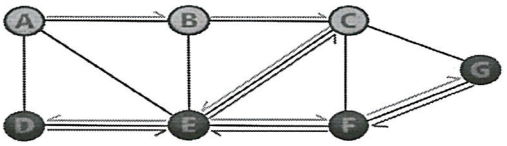
- There is no new vertiex to be visited from F. So use back track.
- Pop F from the Stack.

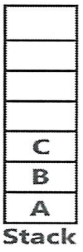




Step 11:

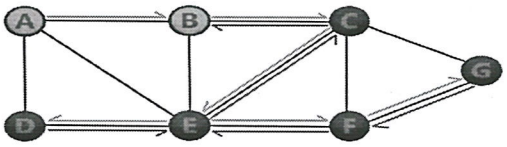
- There is no new vertiex to be visited from E. So use back track.
- Pop E from the Stack.

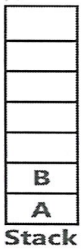




Step 12:

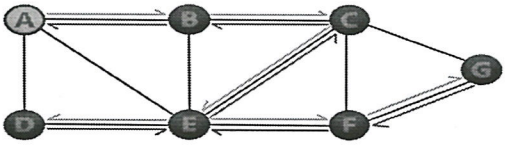
- There is no new vertiex to be visited from C. So use back track.
- Pop C from the Stack.





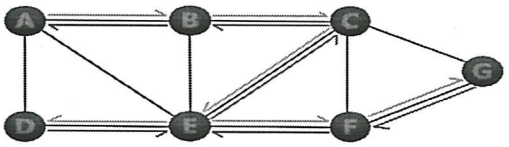
Step 13:

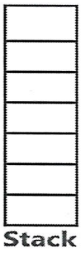
- There is no new vertiex to be visited from B. So use back track.
- Pop B from the Stack.



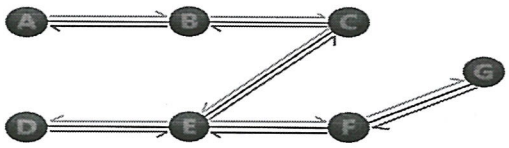
Step 14:

- There is no new vertiex to be visited from A. So use back track.
- Pop A from the Stack.





- Stack became Empty. So stop DFS Treversal.
- Final result of DFS traversal is following spanning tree.



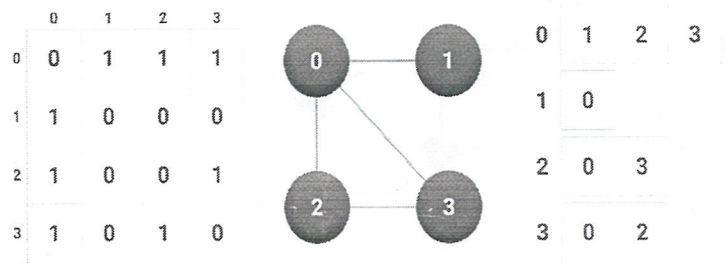
1

c) Explain graph representation methods with example.

Answer:

Adjacency List: An Adjacency list is an array consisting of the address of all the linked lists. The first node of the linked list represents the vertex and the remaining lists connected to this node represents the vertices to which this node

2

	is connected. This representation can also be used to represent a weighted graph. The linked list can slightly be changed to even store the weight of the edge. **Adjacency Matrix**: Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.  	2
Q.5	**Solve any one.**	
a)	Sort following elements using insertion sort. Show all passes. 50, 20, 30, 55, 12, 8, 99, 89 Write best, average, worst case time complexity for insertion sort, Also write algorithm insertion sort. **Answer:** insertion sort: 50 20 30 55 12 8 99 89 temp = 20 20 50 30 55 12 8 99 89 temp = 30 20 30 50 55 12 8 99 89 temp = 55 20 30 50 55 12 8 99 89 temp = 12 12 20 30 50 55 8 99 89 temp = 8 8 12 20 30 50 55 99 89 temp = 99 8 12 20 30 50 55 89 99 temp = 89 Insertion sort complexity: Best case: O(n) Average case: $O(n^2)$ worst case: $O(n^2)$	2 2 2 2
b)	Explain binary search with following example. Element to be search is 50. 8, 12, 20, 25, 30, 40,45, 48, 50, 100 Write algorithm for binary search. **Answer:** 8, 12, 20, 25, 30, 40,45, 48, 50, 100 Start= 0 end = 9 mid=(0+9)/2=4 key=50 A[mid]<key so we search in higher list Start= 5 end = 9 mid=(5+9)/2=7 key=50	2

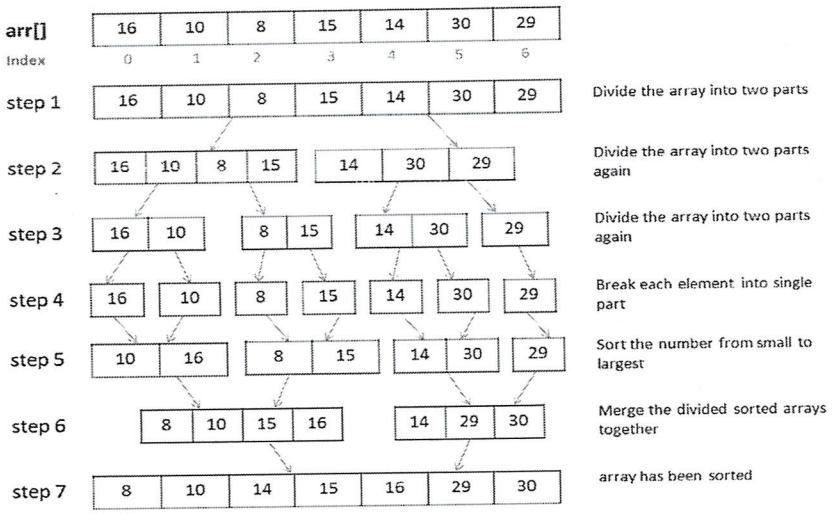
	A[mid]<key so we search in higher list Start= 8 end = 9 mid=(8+9)/2=8 key=50 A[mid]=key so we search element successfully Algorithm for binary search. 1. Sort the array in ascending order. 2. Set the low index to the first element of the array and the high index to the last element. 3. Set the middle index to the average of the low and high indices. 4. If the element at the middle index is the target element, return the middle index. 5. If the target element is less than the element at the middle index, set the high index to the middle index – 1. 6. If the target element is greater than the element at the middle index, set the low index to the middle index + 1. 7. Repeat steps 3-6 until the element is found or it is clear that the element is not present in the array.	2 2 2
Q.6	**Solve any two.**	

a) What is mean by divide and conquer? Explain merge sort with example.

Answer:

Divide and conquer: A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. — 1

Merge sort is a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array. — 1



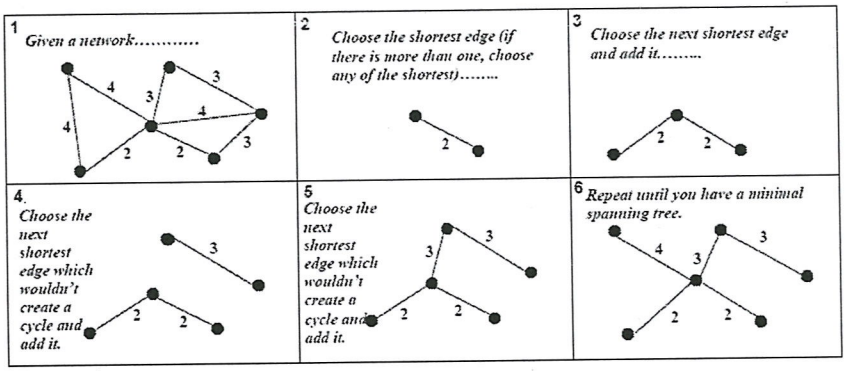
1

b) Discuss Kruskal algorithm with example.

Answer:

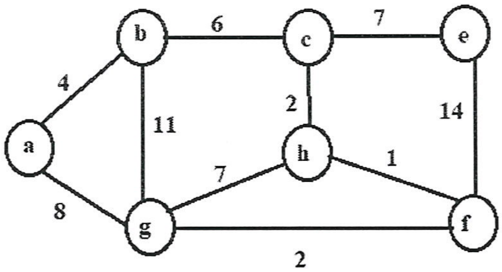
Steps for finding MST using Kruskal's algorithm: — 2

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are (V-1) edges in the spanning tree.

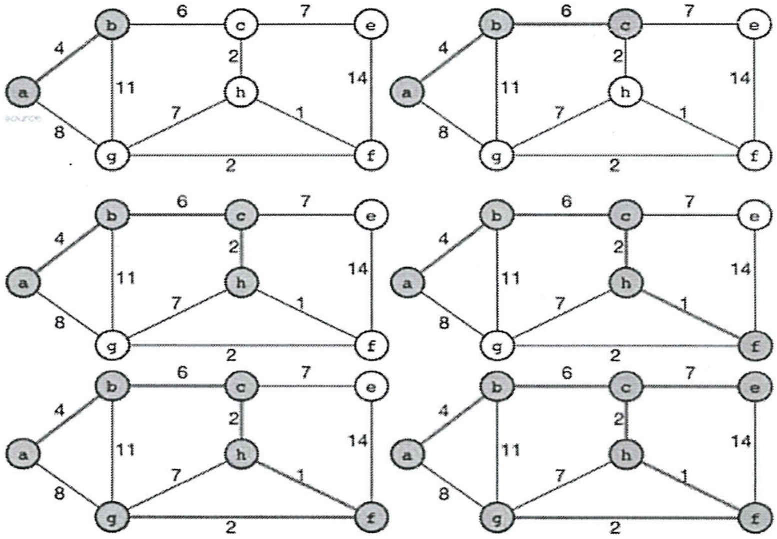


2

c)	Discuss Prims algorithm and Find MST for following graph using Prims algorithm. Consider "a" as source vertex.	



Answer: The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree. 1



1

1

1