FORMAT-1

Semester: 3rd
Programme: B.Tech
Branch/Specialization: CSE

AUTUMN END SEMESTER EXAMINATION-2022

3rd Semester, BTech (Course)

SUBJECT DSA

CODE(CS 2001)

(For ……2021…….. Admitted Batches)

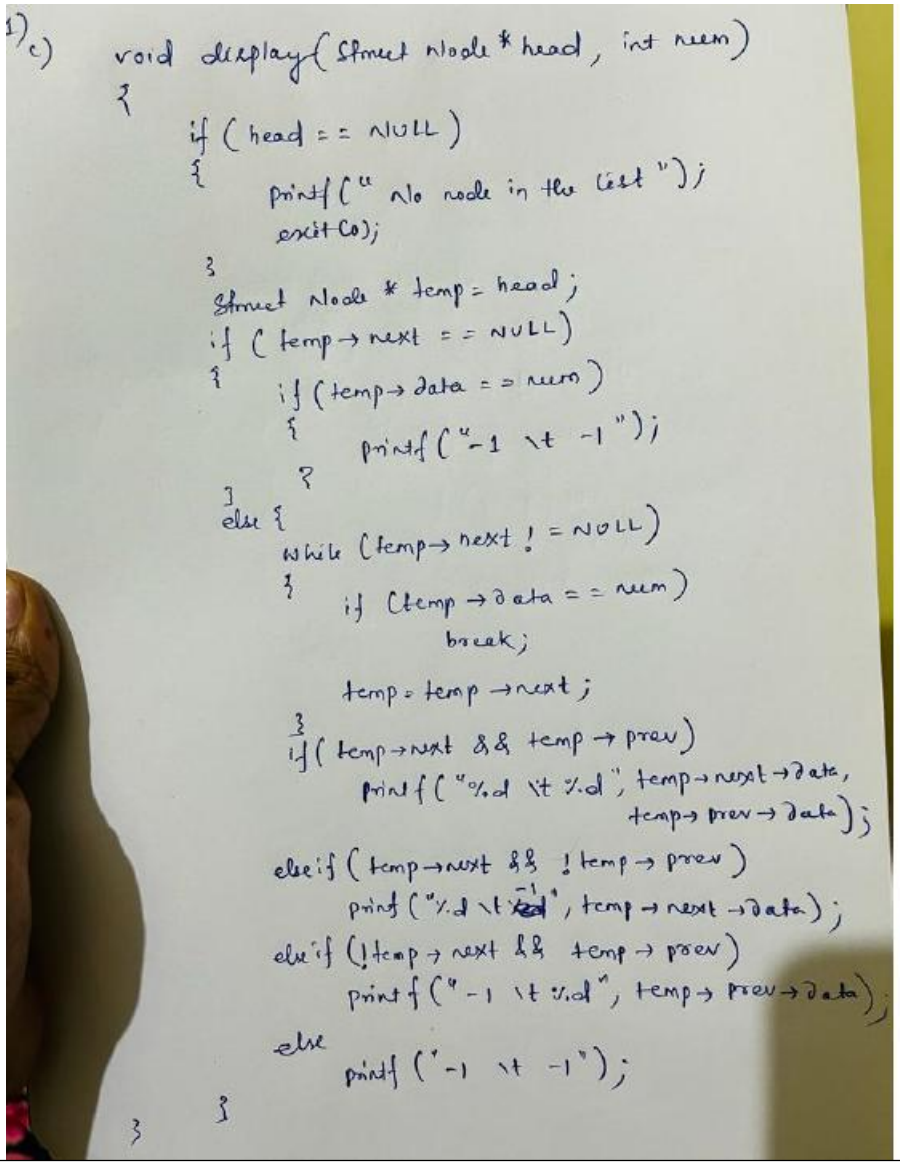
Time: 3 Hours **Full Marks: 50**

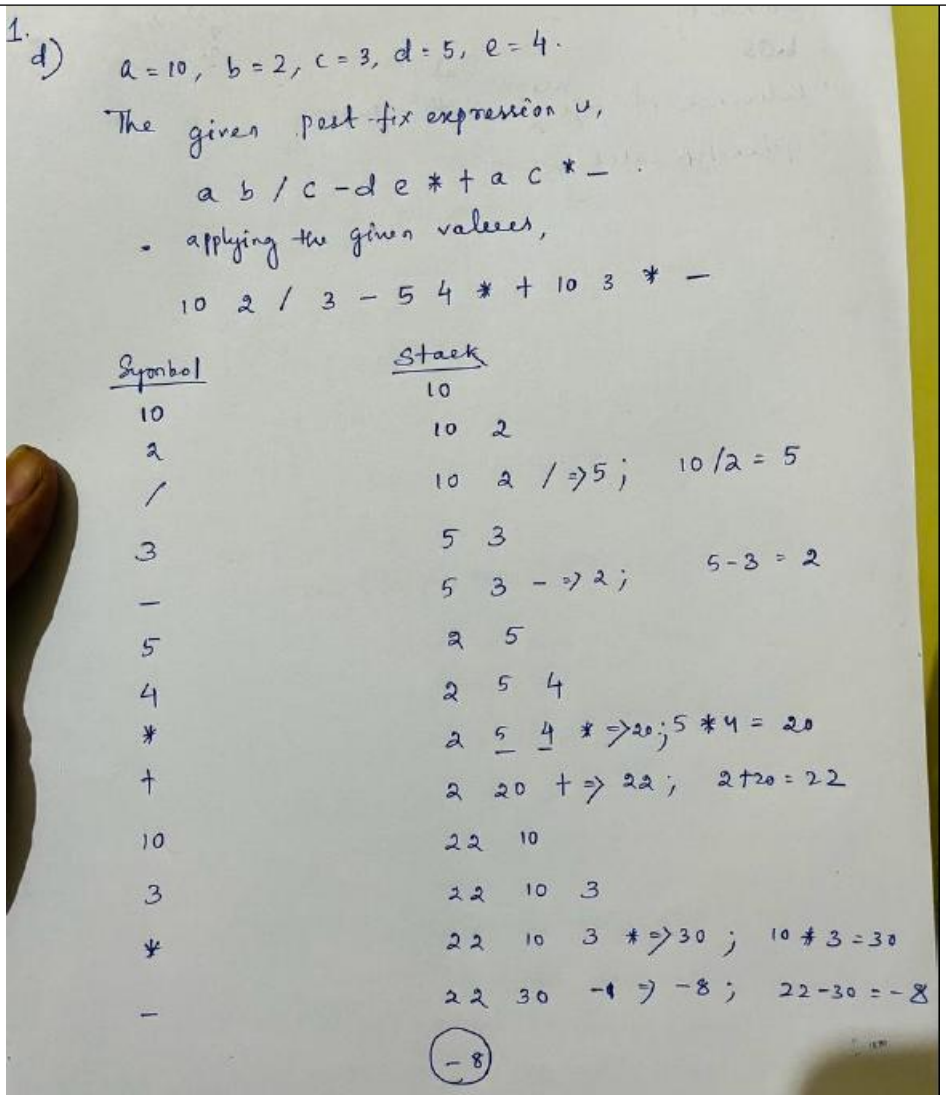
Answer any SIX questions.
Question paper consists of four SECTIONS i.e. A, B, C and D.
Section A is compulsory.
Attempt minimum one question each from Sections B, C, D.
The figures in the margin indicate full marks.
Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

SECTION-A (Learning levels 1 and 2)				Course Outcomes (CO)
1.		Answer the following questions.	[1 × 10]	

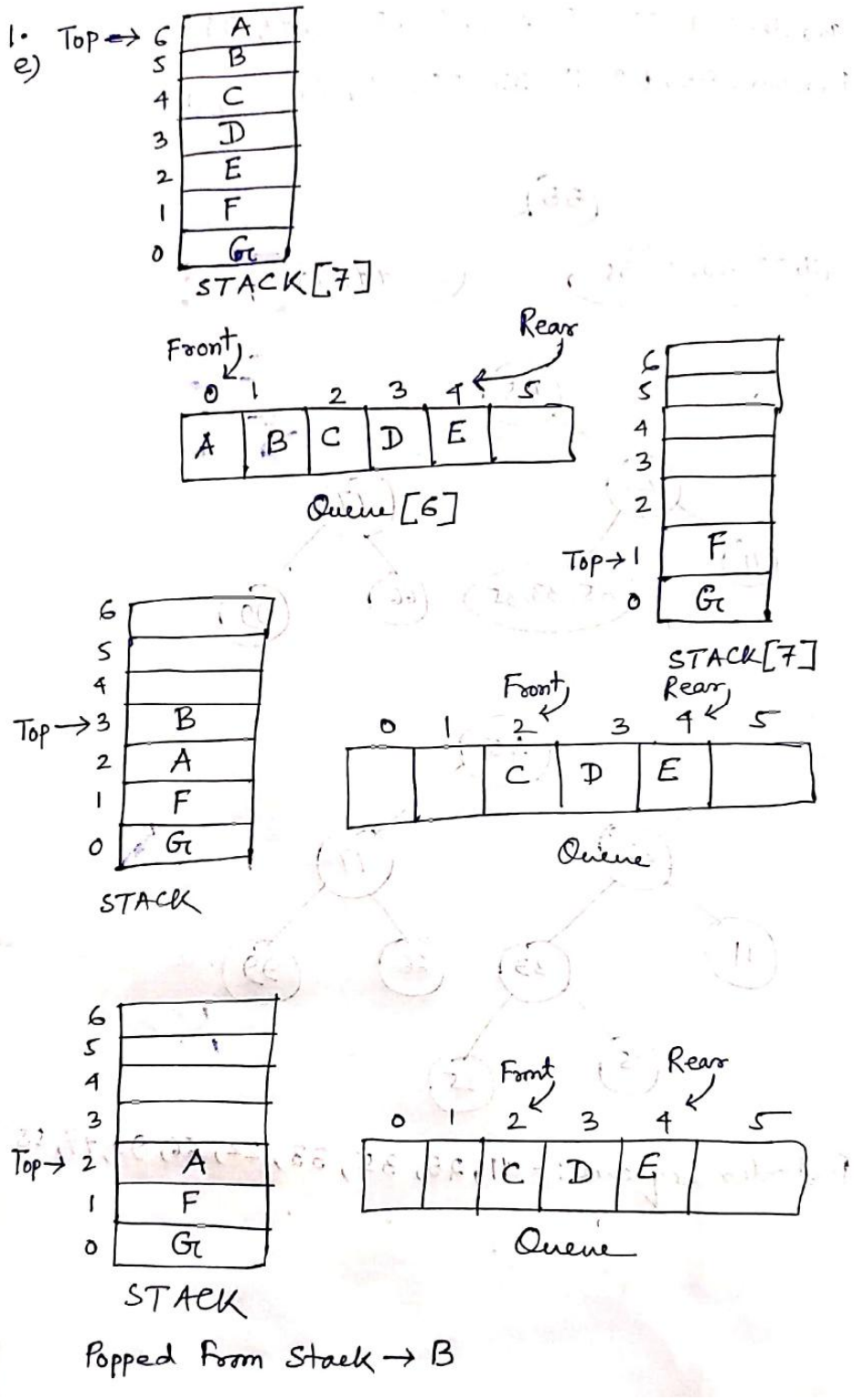
	(a)	Write the content of array p[] for *n=6*.
		int func(int n) {
		int k = 0, p[20];
		for(int i = n/2 ; i<=n; i++)
		for(int j = 1; j < n ; j*=2)
		p[k++] = i +j;
		return k;
		}
		Ans:
		The content of array p[]:
		4 5 7 5 6 8 6 7 9 7 8 10
				CO2

	(b)	Assume that 4-byte of storage is required by each element of the array a[-10..10, -10..10] with starting location 1000. Compute the address of a[6, 5] in row major and column major order. Ans: The given values are: Base Address (BA)= 1000, Each element storage size (w)=4 byte, i= 6, j= 5, Lr=-10, Lc=-10, Total no. of columns (n)= (10-(-10) +1) = 21, Total no. of rows (m) = (10-(-10) +1) = 21 **Row-Major:** Address of A[i][j]= BA + w * [n * (i–Lr) + (j–Lc)] = 1000 + 4 *[21* (6- (-10) + (5- (-10)))] = 1000 + 4 * [21 * 16 + 15] = 1000 + 4 * [336 + 15] = 1000 + 1404 = 2404 **Column-Major:** Address of A[i][j]= BA + w * [(i–Lr) + m * (j–Lc)] = 1000 + 4 *[(6- (-10) + 21 * (5- (-10)))]		CO1

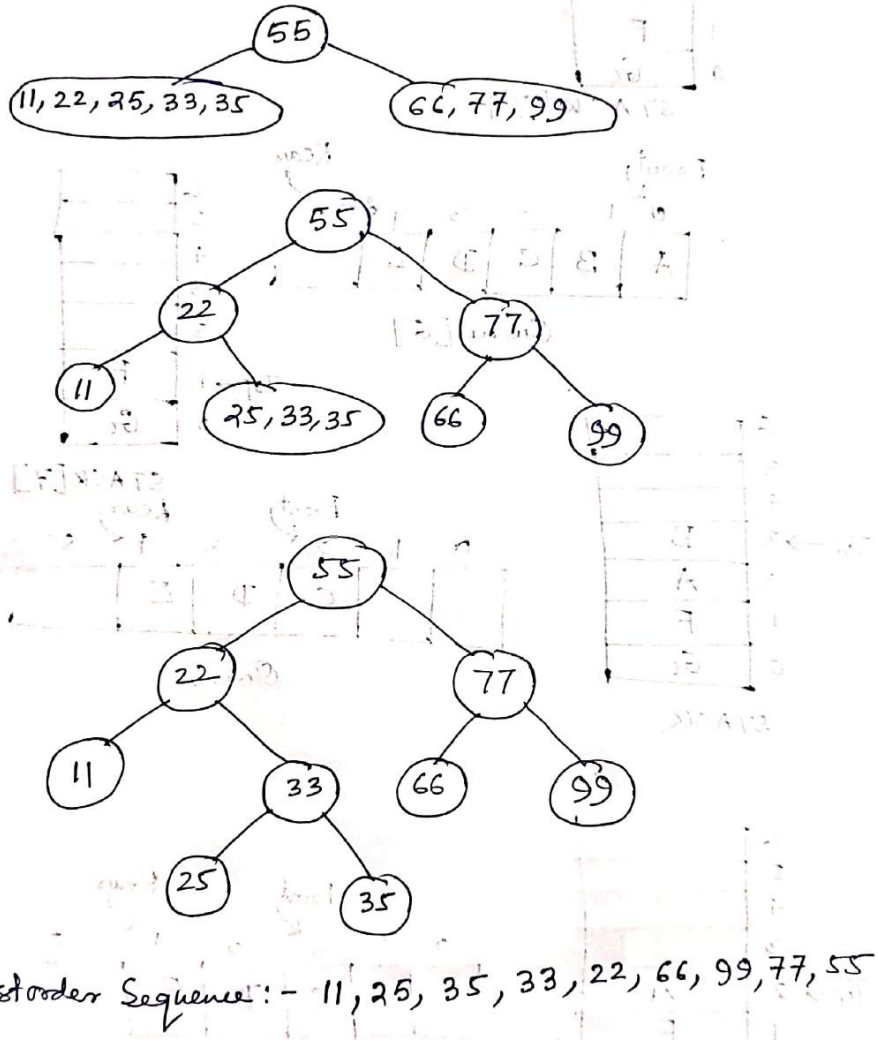
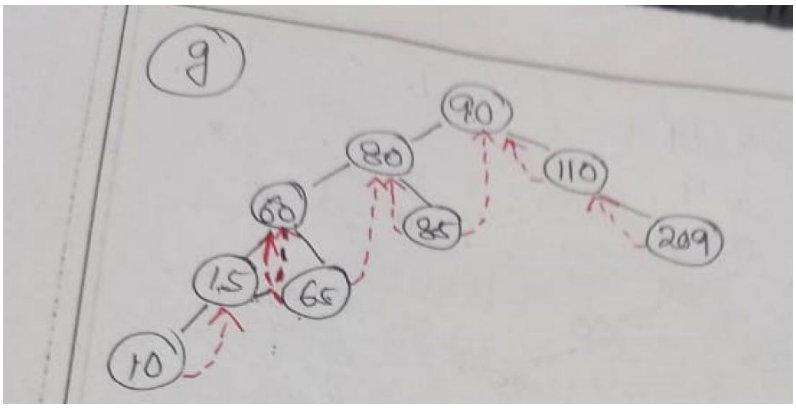
		= 1000 + 4 * [16 + 21 * 15] = 1000 + 4 * 331 = 1000 + 1324 = 2324		
	(c)	Create a function that searches for an integer num in a double linked list and displays the content (info/data) of its left node and right node. In case of unavailability, the content should be displayed as -1. Ans: 		CO 3
	(d)	Evaluate the following postfix expression where a=10, b=2, c=3, d=5, e=4 a b / c - d e * + a c * -		CO 4

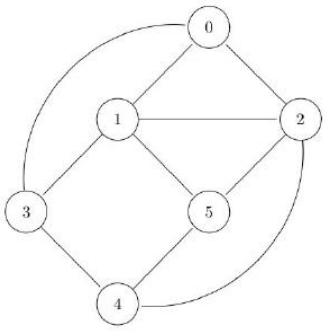
		1. d) a = 10, b = 2, c = 3, d = 5, e = 4. The given post-fix expression is, a b / c - d e * + a c * - - applying the given values, 10 2 / 3 - 5 4 * + 10 3 * -  Symbol / Stack 10 / 10 2 / 10 2 / / 10 2 / => 5; 10/2 = 5 3 / 5 3 - / 5 3 - => 2; 5-3 = 2 5 / 2 5 4 / 2 5 4 * / 2 5 4 * => 20; 5*4 = 20 + / 2 20 + => 22; 2+20 = 22 10 / 22 10 3 / 22 10 3 * / 22 10 3 * => 30; 10*3 = 30 - / 22 30 - => -8; 22-30 = -8 -8		
	(e)	The seven elements A, B, C, D, E, F, and G are pushed onto a stack in reverse order, i.e., starting from G. The stack is popped five times and each element is inserted into a queue. Two elements are deleted from the queue and pushed back onto the stack. Finally, one element is popped from the stack. Stepwise show the content of both stack and queue.		CO 4

Ans:



	(f)	Consider the following inorder and preorder traversal of one binary tree. Construct the tree and find the postorder sequence of the corresponding binary tree. Inorder : 11, 22, 25, 33, 35, 55, 66, 77, 99 Preorder : 55, 22, 11, 33, 25, 35, 77, 66, 99		CO4

		Ans: (f) Inorder: 11, 22, 25, 33, 35, 55, 66, 77, 99 Preorder: 55, 22, 11, 33, 25, 35, 77, 66, 99  Postorder Sequence:- 11, 25, 35, 33, 22, 66, 99, 77, 55		
	(g)	Construct a BST for the given sequence of elements and convert it to the two-way threaded BST tree. 90, 80, 110, 60, 85, 209, 15, 65, 10 Ans:  		CO4, CO6
	(h)	Write/draw the linked list representation and array representation of the following graph.		Co4,Co 5,Co6

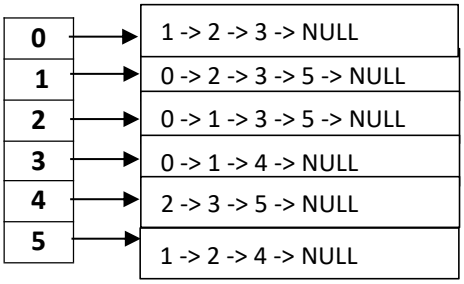


Ans:

Array Representation of the graph (**Adjacency Matrix**)

	0	**1**	**2**	**3**	**4**	**5**
0	0	1	1	1	0	0
1	1	0	1	1	0	1
2	1	1	0	0	1	1
3	1	1	0	0	1	0
4	0	0	1	1	0	1
5	0	1	1	0	1	0

Linked List Representation (**Adjacency List**)



(i) Let QuickSort program P is applied on the 2 sets of inputs to sort the numbers in ascending order using the first element as a pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. What is the value of t1 and t2, and what is the relation (< or >) among them? CO 6

Ans:

t1 is the total number of comparison made by P for the input { 1,2,3,4,5} which is O(logn).

t2 is the total number of comparison made by P for the input { 4,1,5,3,2} which is $O(n^2)$.

So , t1 > t2 .

(j) Write a function that concatenates two circular linked lists into one circular linked list. CO6

Ans:

```
struct node *concat( struct node *tail1, struct node *tail2) {
        struct node *ptr;
        if(tail1==NULL) {
                tail1=tail2;
                return tail1;
        }
        if(tail2==NULL )
                return tail1;
        ptr=tail1->next;
        tail1->next=tail2->next;
        tail2->next=ptr;
        tail1=tail2;
```

```
        return tail1;
}
```

SECTION-B (Learning levels 1,2, and 3)

Course Outcomes (CO)

2. (a) Write a C-code or pseudo-code to separate all odd numbers and even numbers of a list of *n* elements such that all odd numbers will appear before the even numbers. Choose a suitable data structure for the list. For example, consider the following input and output of the program. [4] CO6

Input: 11, 22, 33, 44, 66, 55, 77, 88, 99, 10
Output: 11, 33, 55, 77, 99, 22, 44, 66, 88, 10

Ans:

```
struct node {
        int num;
        struct node *next;
};
void generate_oddeven(struct node *list, struct node **head) {
        struct node *even = NULL, *odd = NULL, *temp;
        struct node *reven, *rodd;
        while (list != NULL) {
                temp = (struct node *)malloc(sizeof(struct node));
                temp->num = list->num;
                temp->next = NULL;
                if (list->num % 2 == 1) {
                        if (odd == NULL)
                                odd = temp;
                        else
                                rodd->next = temp;
                        rodd = temp;
                }
                else {
                        if (even == NULL)
                                even = temp;
                        else
                                reven->next = temp;
                        reven = temp;
                }
                list = list->next;
        }
        rodd->next = even;
        *head = odd;
}
```

(b) Write insertion and deletion functions to implement an output restricted double-ended queue. [4] CO4

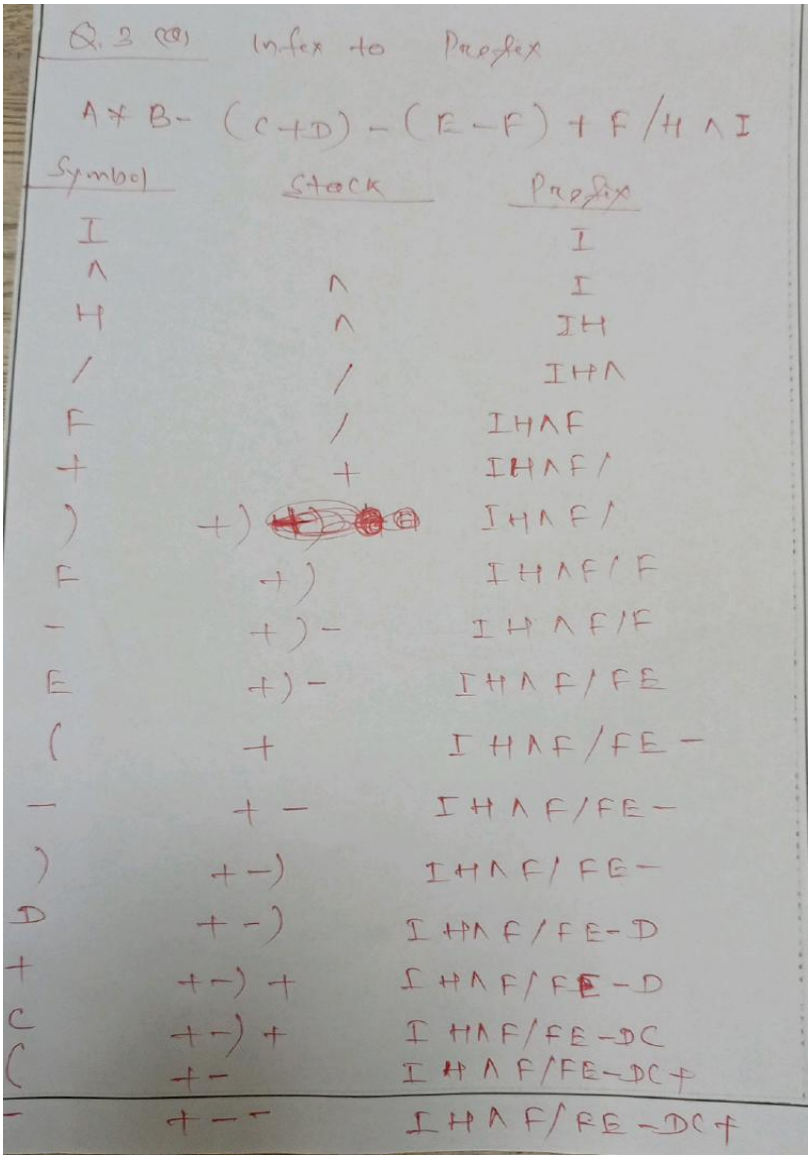
Ans:

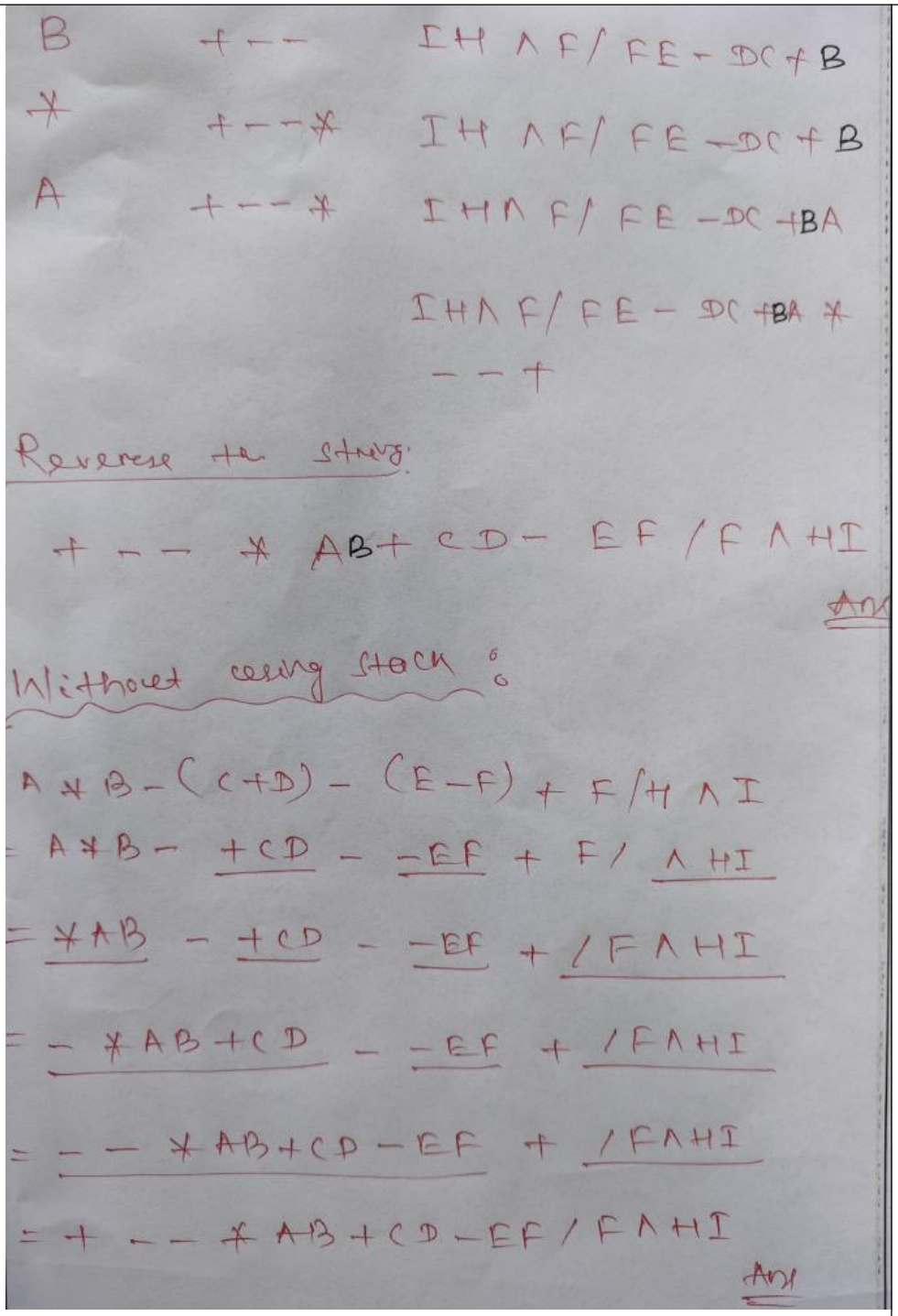
```
void insertAtrear(int val ) {
        if ( (front == 0 && rear == MAX-1 ) || (front == rear+1) ) {
                printf(" OVERFLOW…");
                return;
        }
        if (front == -1)  {
                front = 0;
                rear = 0;
        }
        else  {
                if (rear == MAX-1)
```

```
                    rear = 0;
                else
                    rear = rear+1;
        }
        deque[rear] = val ;
}

void insertAtfront(int val ) {
        if( (front ==0 && rear == MAX-1) || (front == rear+1) ) {
                printf(" OVERFLOW…");
                return;
        }
        if (front == -1) {
                front = 0;
                rear = 0;
        }
        else {
                if(front == 0)
                        front = MAX - 1 ;
                else
                         front = front - 1 ;
        }
        deque[front] = val;
}

void deleteAtfront( ) {
        if ( front == -1 )
        {
                printf("  UNDERFLOW…");
                return ;
        }
        printf("  The deleted element is :  %d", deque[front]);
        if (front == rear)        {
                front = -1 ;
                rear = -1 ;
        }
        else {
                if ( front == MAX - 1 )
                        front = 0;
                else
                        front = front+1;
        }
}
```

3.	(a)	Convert the following infix expression to its corresponding prefix expression using the suitable data structure. A * B - (C + D) - (E - F) + F / H ^ I	[4]	CO4

Ans:


Q.3 (a) Infix to Prefix
A * B - (C + D) - (E - F) + F / H ^ I
Symbol Stack Prefix
I I
^ ^ I
H ^ IH
/ / IH^
F / IH^F
+ + IH^F/
) +) IH^F/
F +) IH^F/F
- +)- IH^F/F
E +)- IH^F/FE
(+ IH^F/FE-
- +- IH^F/FE-
) +-) IH^F/FE-
D +-) IH^F/FE-D
+ +-) + IH^F/FE-D
C +-) + IH^F/FE-DC
(+- IH^F/FE-DC+
- +-- IH^F/FE-DC+

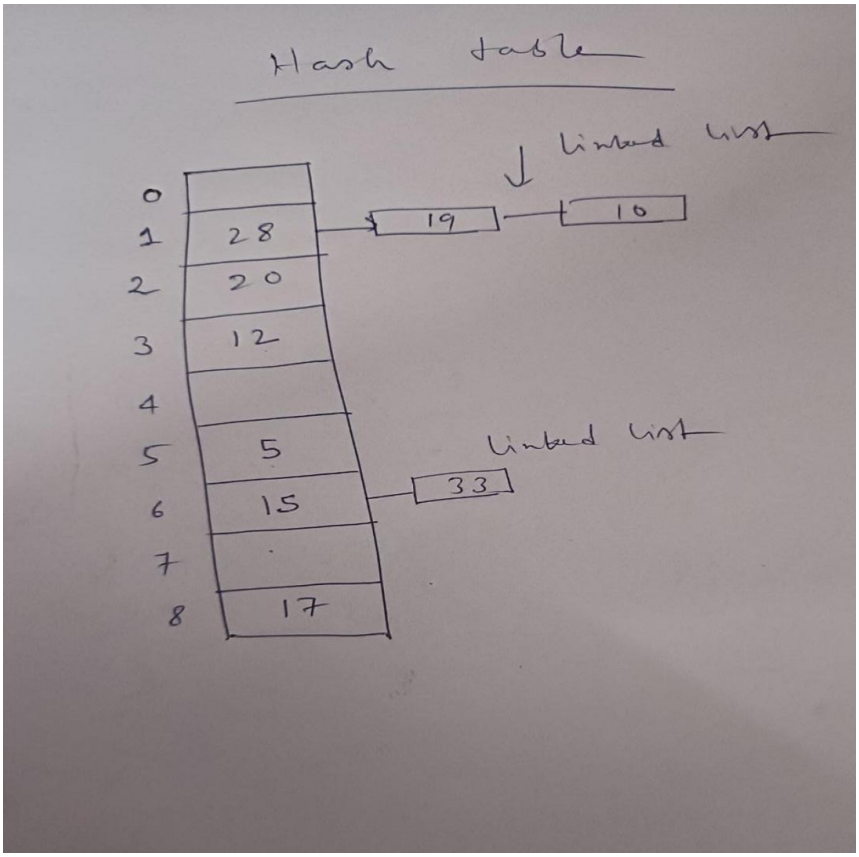
				
	(b)	What is the requirement of hashing? Demonstrate insertion of keys 5, 28, 19, 15, 20, 33, 12, 17, and 10 into a hash table with separate chaining-based collision resolution strategy. Let the table have 9 slots and it uses the division hash function. Ans: Hashing is a concept that is used to search an item in O(1) time in a hash table.Hash table is a data structure in which keys are mapped to array positions by a hash function.A hash function is a mathematical formula which, when applied to a key, produces an integer which can be used as an index for the key in the hash table. The main aim of a hash function is that elements should be relatively, randomly, and uniformly distributed. It produces a unique set of integers within some suitable range in order to reduce the number of collisions.	[4]	CO5

Given keys : 5, 28, 19, 15, 20, 33, 12, 17, 10

$m = 9$.

Division hash function $h(x) = x \bmod m$

keys	h(x)
5	5 % 9 = 5
28	28 % 9 = 1
19	19 % 9 = 1
15	15 % 9 = 6
20	20 % 9 = 2
33	33 % 9 = 6
12	12 % 9 = 3
17	17 % 9 = 8
10	10 % 9 = 1



		SECTION-C (Learning Levels 3 and 4)		Course Outcomes (CO)

4.	(a)	Given two single linked lists, write the code to merge their nodes to make one list, taking nodes alternately between the two lists. So the output with {1, 2, 3} and {7, 13, 1} should yield {1, 7, 2, 13, 3, 1}. If either list runs out, all the nodes should be taken from the other list.
		
		Ans:
		void merge(struct Node *p, struct Node *q) {
		struct Node *p_curr = p, *q_curr = q;
		struct Node *p_next, *q_next, *t_curr;
		while (p_curr != NULL && q_curr != NULL) {
		p_next = p_curr->next;
		q_next = q_curr->next;
		q_curr->next = p_next;
		p_curr->next = q_curr;
		p_curr = p_next;
		t_curr- = q_curr;
		q_curr = q_next;
		}
		if(p_curr!=NULL)
		t_curr = p_curr;
		if(q_curr!=NULL)
		t_curr = q_curr;
		}
			[4]	CO6

	(b)	Perform the ascending order sorting of the given elements by using the Heap sort algorithm. Illustrate your approach step by step. 20, 30, 35, 45, 57, 11 Ans:	[4]	C05

4. (b) Ascending order Sorting using Heap sort Algo—

Input:- | 20 | 30 | 35 | 45 | 57 | 11 |

Step 1: Build a complete binary tree (Fig 1)

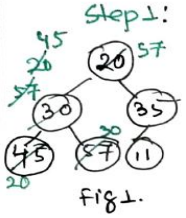
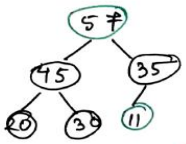


fig 1.

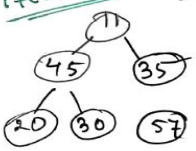
Step 2: Convert the tree to a Max-Heap using Heapify method.



Step 3: For i ← length_heap(A) to 2
do exchange A[1] ↔ A[i]
heap_size(A) ← heap_size(A) − 1;
Call Max-heap(A, 1);

Array A (Heap)

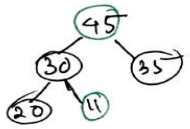
Iterations



| 11 | 40 | 35 | 20 | 30 | 57 |

placed in a proper cell

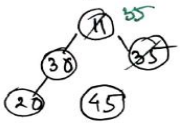
<Heapify>



| 45 | 30 | 35 | 20 | 11 | 57 |

N:B:— For last 2 unsorted elements only exchange produces the desired result

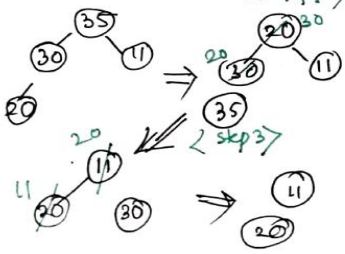
<exchange last node to first & Apply Heapify>



| 35 | 30 | 11 | 20 | 45 | 57 |

Last two element Sorted

<exchange A[1] to A[i] & Apply Heapify>



<Step 3>

| 30 | 20 | 11 | 35 | 45 | 57 |

sorted

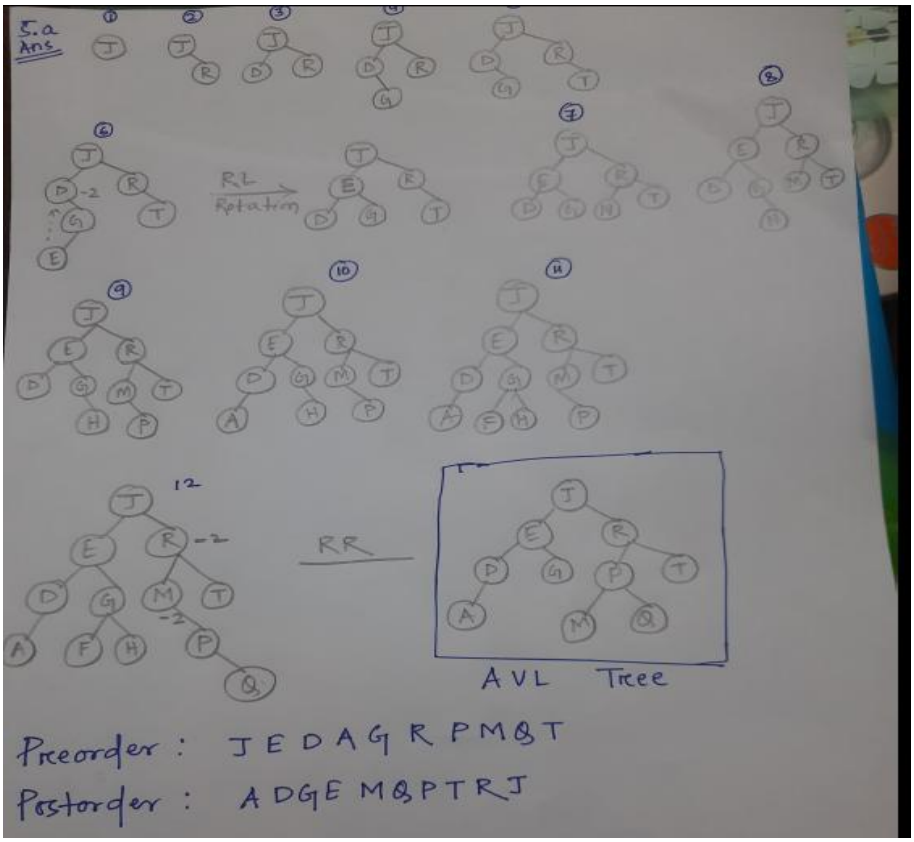
| 20 | 11 | 30 | 35 | 45 | 57 |

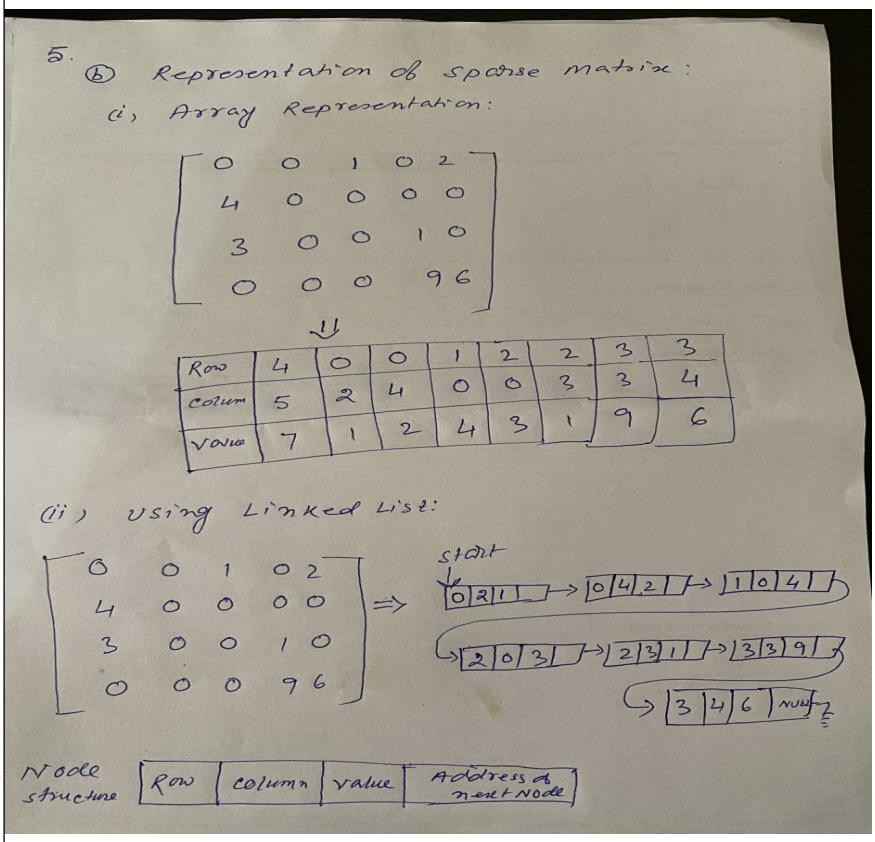
sorted

| 11 | 20 | 30 | 35 | 45 | 57 |

Sorted elements

5.	(a)	Insert the following sequence of characters in the height-balanced BST (AVL tree). Find the preorder and postorder traversal of the AVL tree. J, R, D, G, T, E, M, H, P, A, F, Q Ans:	[4]	C04

				
	(b)	Represent the following sparse matrix in the triplet form using the data structure array and linked list. Write the C-code or pseudo-code to add two sparse matrices using any of the representation. $\begin{bmatrix} 0 & 0 & 1 & 0 & 2 \\ 4 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 9 & 6 \end{bmatrix}$ Ans:	[4]	CO3



C code to add two sparse matrix

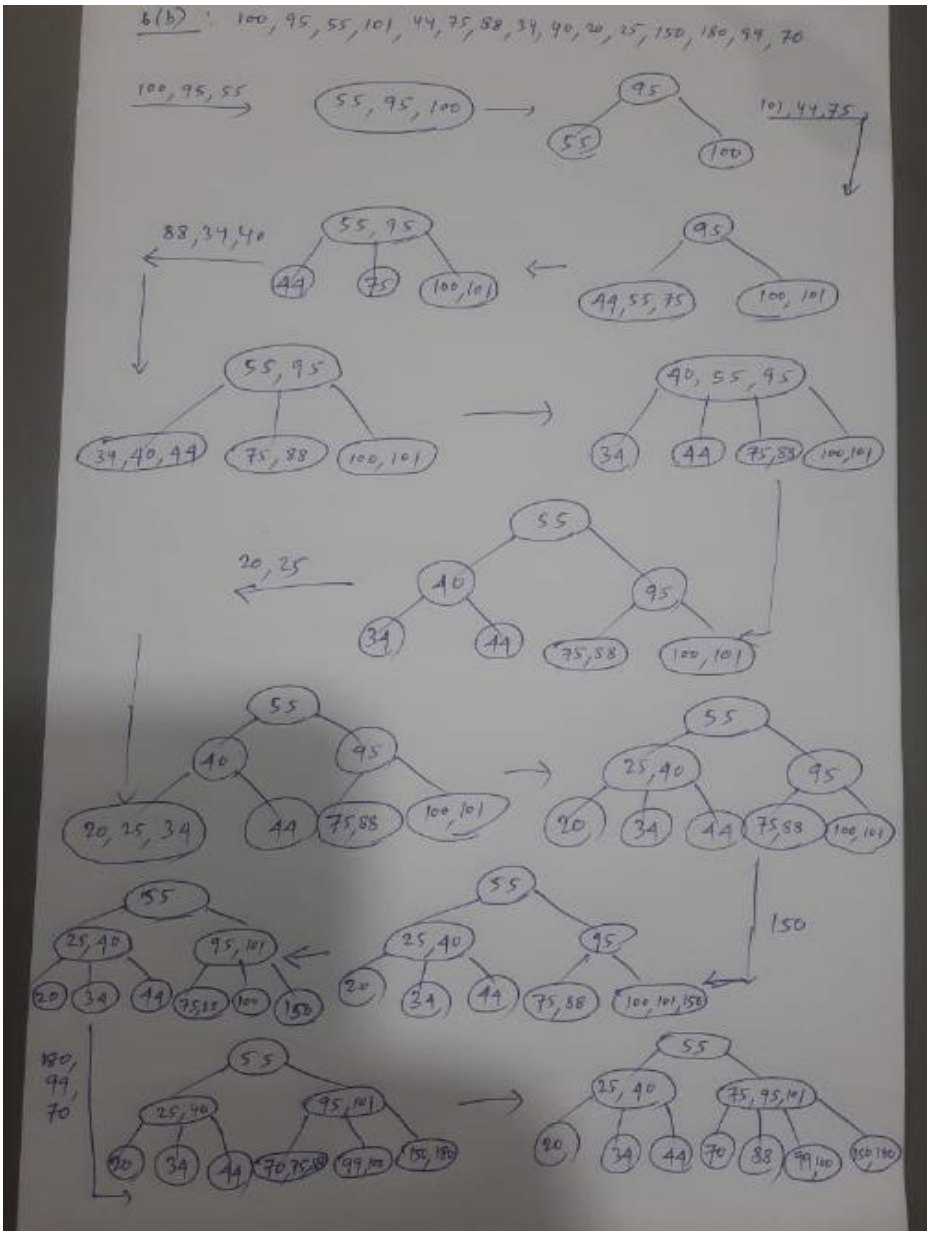
```
#include<stdio.h>
int main() {
   int
i,j,n,c1=1,c2=1,count=1,a[10][10],b[10][10],s[20][20],t[20][20],sum[50][50],last[
10][10];
   printf("\nEnter dimension:");
   scanf("%d",&n);
   printf("\nEnter elements of 1st matrix:");
   for(i=1;i<=n;i++)   {
      for(j=1;j<=n;j++)
         scanf("%d",&a[i][j]);
   }
   printf("\nEnter elements of 2nd matrix:");
   for(i=1;i<=n;i++)   {
      for(j=1;j<=n;j++)
         scanf("%d",&b[i][j]);
   }
   for(i=1;i<=n;i++)  {
      for(j=1;j<=n;j++)   {
            if (a[i][j]==0)
               continue;
            else   {
               s[c1][1]=i;
               s[c1][2]=j;
               s[c1][3]=a[i][j];
               c1++;

            }
         }
   }
   for(i=1;i<=n;i++)  {
```

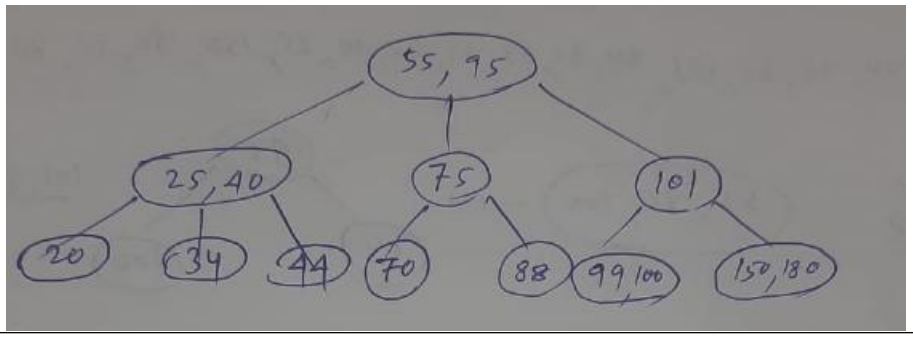
```
      for(j=1;j<=n;j++)  {
            if (b[i][j]==0)
               continue;
            else   {
               t[c2][1]=i;
               t[c2][2]=j;
               t[c2][3]=b[i][j];
               c2++;
            }
         }
   }
   printf("\nThe sparse matrix form of A is:\n");
   for(i=1;i<c1;i++)  {
      for(j=1;j<=3;j++)
         printf("%d\t",s[i][j]);
      printf("\n");
   }
   printf("\nThe sparse matrix form of B is:\n");
   for(i=1;i<c2;i++)   {
      for(j=1;j<=3;j++)
         printf("%d\t",t[i][j]);
      printf("\n");
   }
for(i=1;i<c1;i++) {
   for(j=1;j<c2;j++)   {
      if ((s[i][1]==t[j][1])&&(s[i][2]==t[j][2]))    {
      Sum[count][1]=s[i][1];
      sum[count][2]=s[i][2];
      sum[count][3]=s[i][3]+t[j][3];
      s[i][3]=0;
      t[j][3]=0;
      count++;
      }
      }
}
for(i=1;i<c1;i++)  {
   if(s[i][3]!=0)   {
      sum[count][1]=s[i][1];
      sum[count][2]=s[i][2];
      sum[count][3]=s[i][3];
      count++;
  }
for(i=1;i<c2;i++) {
   if(t[i][3]!=0)  {
      sum[count][1]=t[i][1];
      sum[count][2]=t[i][2];
      sum[count][3]=t[i][3];
      count++;
      }
}
printf("\nThe sparse matrix form of sum is:\n");
   for(i=1;i<count;i++)  {
      for(j=1;j<=3;j++)
         printf("%d\t",sum[i][j]);
      printf("\n");
   }

for(i=1;i<=n;i++)  {
```

```
    for(j=1;j<=n;j++)
      last[i][j]=0;
  }
  for(i=1;i<count;i++)  {
    last[sum[i][1]][sum[i][2]]=sum[i][3];
  }
  printf("\nSum is:\n");
  for(i=1;i<=n;i++)  {
    for(j=1;j<=n;j++)
      printf("%d\t",last[i][j]);
    printf("\n");
  }
}
```

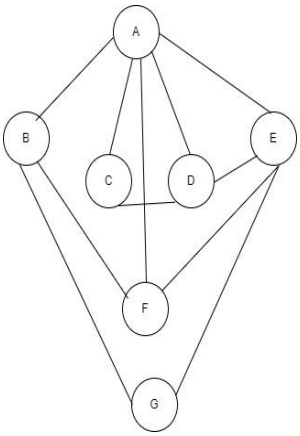
6.	(a)	Write the appropriate iterative procedure to traverse a binary search tree that outputs the elements in an arranged (ascending) form. In-order traversal will produce the outputs of the elements in an ascending order. Hence, the iterative procedure/algorithm for in-order traversal procedure is given below. Ans: *Algorithm* 1. Initialize an empty stack. 2. Push the current node (starting from the root node) onto the stack. Continue pushing nodes to the left of the current node until a NULL value is reached. 3. If the current node is NULL and the stack is not empty: a. Remove and print the last item from the stack. b. Set the current node to be the node to the right of the removed node. c. Repeat the second step of the algorithm. 4. If the current node is NULL and the stack is empty, then the algorithm has finished.	[4]	CO4
	(b)	Perform insertion of the following elements in a 3-way B-tree step by step. 100, 95, 55, 101, 44, 75, 88, 34, 40, 20, 25, 150, 180, 99, 70	[4]	CO4

Ans:



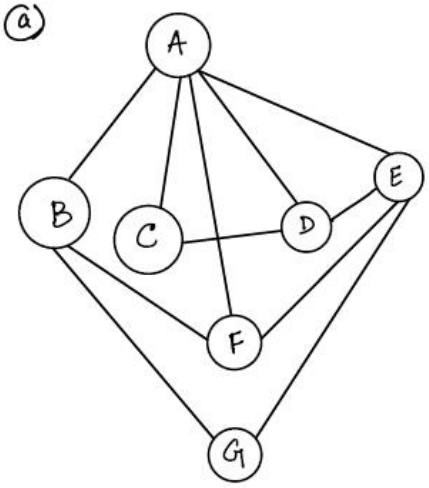


		SECTION-D (Learning levels 4,5,6)		Course Outcomes (CO)

7.	(a)	Write an appropriate traversal algorithm so that the algorithm visits the vertices in a depth-wise order starting from a given node. Illustrate the step by step traversal considering starting vertex as vertex 'A'. 	[4]	CO4, CO 6

Ans:

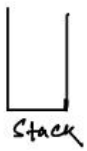
Q.7 (a)



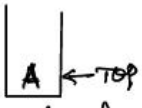

```
Procedure DFS (V)
{
  visited (V) = TRUE
  for each vertex 'w'
  adjacent to 'V'
  { if NOT visited (w)
    {
      call DFS(w)
    }
  }
}
```

Considering A as the starting vertex the DFS Traversal is

(1) Initialize an empty stack

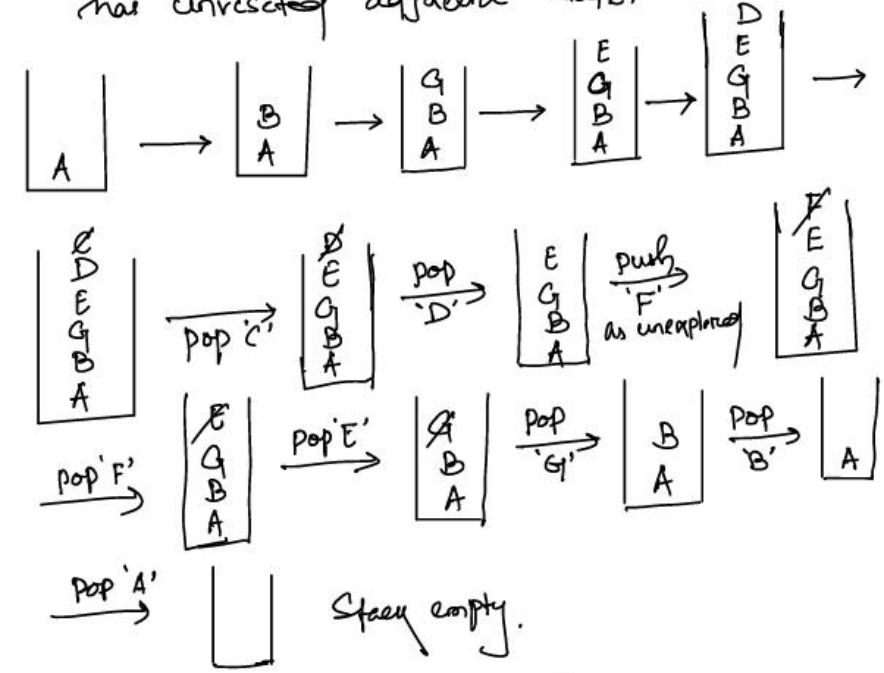


(2) Mark Node A as visited and put it into the stack

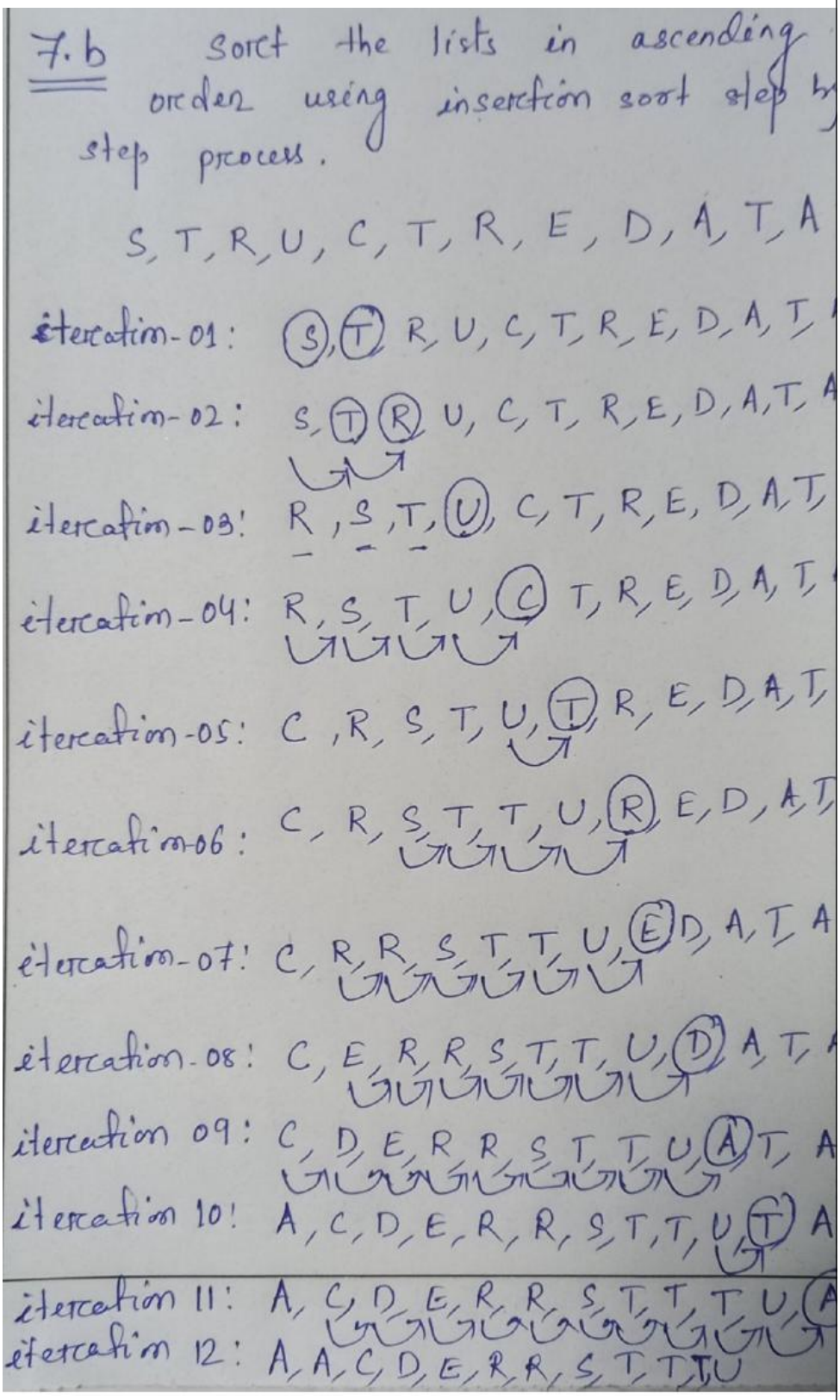


(3) Explore any unvisited adjacent Node from A. Here we have four Nodes (B, C, D, E). We can pick any one of them and explored it.

(4) Mark B as visited and put it onto the stack. Explore any unvisited adjacent Node from B. Both F & G are the unvisited Node. We can pick any one of them and explore it.

		⑤ when a Node doesnot have any unvisited adjacent Node pop it from the stack, we keep poping until we find a Node that has unvisited adjacent Node.  other possible Sequences are: 2) A C D E G B F 3) A D C E F B G 4) A E D C F B G		
	(b)	Sort the following lists in ascending order using insertion sort. Show the step by step process. S, T, R, U, C, T, U, R, E, S, D, A, T, A	[4]	CO5, CO6

Ans:



8.	(a)	Write a program to insert n records (nodes) of student profiles in a double linked list such that the records or nodes will be present in ascending order of the student's CGPA. The student profile consists of roll number(int), name (string), and CGPA (float). Ans: #include<stdio.h> #include<stdlib.h>	[4]	CO3,CO 6

```
struct node {
        struct node *prev;
        int roll;
        char name[10];
        float CGPA;
        struct node *link;
} *head=NULL;

void create();
void sort();
void display();
int main() {
        int ch; char choice;
        do {
                printf("----------------------\n");
                printf("1. Create\n");
                printf("2. Sort\n");
                printf("3. Display\n");
                printf("----------------------\n");
                printf("enter your choice:\n");
                scanf("%d",&ch);
                switch(ch) {
                        case 1: create(); break;
                        case 2: sort(); break;
                        case 3: display(); break;
                        default: printf("Invalid choice");
                }
                printf("\nwant options?");
                scanf(" %c",&choice);
        }
        while (choice=='y'||choice=='Y');
        return 0;
}

void create() {
        struct node *newnode,*current;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("enter data of newnode- Roll,name,CGPA :");
        scanf("%d",&newnode->roll);
        scanf("%s",newnode->name);
        scanf("%f",&newnode->CGPA);
        newnode->prev=NULL;
        newnode->link=NULL;
        if(head==NULL)
                head=newnode;
        else {
                current=head;
                while(current->link!=NULL)
                        current=current->link;
                newnode->prev=current;
                current->link=newnode;
        }
}

void sort() {
        struct node *current=NULL,*ptr=NULL;
        int temp;
```

```
        current=head;
        if(head==NULL)
                return;
        else {
                for(current=head;current->link!=NULL;current=current-
>link) {
                        for( ptr=current->link;ptr!=NULL;ptr=ptr->link) {
                                if(current->CGPA>ptr->CGPA) {
                                        temp=current->CGPA;
                                        current->CGPA=ptr->CGPA;
                                        ptr->CGPA=temp;
                                }
                        }
                }
        }
}

void display() {
        struct node *ptr;
        if(head==NULL)
                printf("List is empty");
        else {
                ptr=head;
                while(ptr!=NULL) {
                        printf("%f-->",ptr->CGPA);
                        ptr=ptr->link;
                }
        }
}
```

(b) Write a c-code or pseudo code that will create a new linked list by removing alternative nodes from the given linear linked list. Display the resultant two linked lists (old and new). [4] CO3,CO6

Input: 2->5->3->7->8
Output: Old List: 5->7
New List: 2->3->8

Input: 2->5->3->7
Output: Old List: 5->7
New List: 2->3

Ans:

```
#include <stdio.h>
#include <stdlib.h>

struct LL{
  int data;
  struct LL *next;
};

// For Creation of Link-List
void insertAtBeginning(struct LL**head,int dataToBeInserted)
{
  struct LL*temp=*head;
  struct LL*curr=(struct LL*) malloc(sizeof(struct LL));
     curr->data=dataToBeInserted;
     curr->next=NULL;
  if(*head==NULL)
     *head=curr;

  else
     {
     curr->next=*head;
     *head=curr;
     }


}
//Alternating sublist function
void alternatingSublist(struct LL**head,struct LL ** a,
struct LL ** b)
{
  struct LL * temp = *head , *A = *a , *B = *b;

 //decides if the present node goes into list A or not
  int listA;


  listA = 1;
  while(temp != NULL)
  {
     if(listA)
     {
     if(A == NULL) //if first node then create node
     {
     *a = (struct LL*) malloc(sizeof(struct LL));
     (*a)->data = temp->data;
     (*a)->next = NULL;
     A = *a;
     }

     else //else create the node for next and then move to
the next one
     {
     A->next = (struct LL*) malloc(sizeof(struct LL));
     A = A->next;
     A->data = temp->data;
     A->next = NULL;
     }

     }
     else
     {
     if(B == NULL) //if first node then create node
     {
     *b = (struct LL*) malloc(sizeof(struct LL));
     (*b)->data = temp->data;
     (*b)->next = NULL;
     B = *b;
     }

     else //else create the node for next and then move to
the next one
     {
     B->next = (struct LL*) malloc(sizeof(struct LL));
     B = B->next;
```

```
        B->data = temp->data;
        B->next = NULL;
        }


        }

        listA = !listA ; //alternate lists
        temp = temp->next; // move to next node
    }

}

//For Display
void display(struct LL**head)
{
   struct LL*temp=*head;
   while(temp!=NULL)
      {
      if(temp->next!=NULL)
           printf("%d->",temp->data);
      else
           printf("%d",temp->data);

      temp=temp->next; //move to next node
      }
        printf("\n");
}

//Main
int main()
{

   struct LL *head = NULL; //initial list has no elements
   insertAtBeginning(&head,8);
   insertAtBeginning(&head,7);
   insertAtBeginning(&head,3);


   insertAtBeginning(&head,5);
   insertAtBeginning(&head,2);

   printf("\nCurrent List is :-\n");
   display(&head);

   struct LL * A = NULL , *B = NULL;

   alternatingSublist(&head , &A , &B);
   printf("\nAfter the alternate deletion the new sublists are
:\n");

   printf("\nNewList: ");
   display(&A);


   printf("\nOld List: ");
   display(&B);

   return 0;
}
```

Output:

A.

B.

*****

Satarupa Mohanty

Signature of Paper Moderator