#DS practicle

// 1.(a) Write a program to store the elements in 1-D array and perform the operations like //searching, sorting and reversing the elements. [Menu Driven]

```
#include<stdio.h>
#include<conio.h>
int a[20],n,i,j,key,temp,first,last,middle;
void display();
void search();
void sort();
void reverse();
int main()
{
	int choice;
	clrscr ();
	printf("Enter the size of the array elements:");
	scanf("%d",&n);
	printf("Enter the elements for the array:");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	display();
	do{
		printf("\n\n--------Menu----------\n");
		printf("1.Search\n");
		printf("2.Sort\n");
		printf("3.reverse\n");
		printf("4.Exit\n");
		printf("\nEnter your choice:\t");
```

```
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:search();break;
                        case 2:sort();break;
                        case 3:reverse();break;
                        case 4:exit(0);break;
                        default:printf("\nInvalid choice:\n");break;
                }
         }
        while(choice!=4);
        return 0;
}
void display()
{
        printf("\nThe array elements are:\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
}
void search()
{
        printf("\nEnter the element to be searched:\t");
        scanf("%d",&key);
        first = 0;
        last = n - 1;
        middle = (first+last)/2;
        while (first <= last)
        {
```

```
                if (a[middle] < key)
                        first = middle + 1;
                else if (a[middle] == key)
                {
                        printf("%d found at location %d.\n", key, middle+1);
                         break;
                }
                else
                        last = middle - 1;
                middle = (first + last)/2;
    }
    if (first > last)
        printf("Not found! %d is not present in the list.\n", key);
}
void sort()
{
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-1-i;j++)
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
        }
        printf("\nAfter sorting the array elements are:\n");
        display();
}
void reverse()
```

```
{
        j = i - 1;
        i = 0;
        while (i < j)
        {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
                i++;
                j--;
        }
        printf("reverse of on array in \n");
        display();
}
```

// 1. (b). Read the two arrays from the user and merge them and display the elements in sorted order.[Menu Driven]

```
#include<stdio.h>
#include<conio.h>
int array1[50], array2[50], array3[100], m, n,i,j, k ,temp;
void mergesort();
int main()
{
   int choice;
   clrscr ();
   printf("\n Enter size of array Array 1: ");
   scanf("%d", &m);
   printf("\n Enter sorted elements of array 1: \n");
   for (i = 0; i < m; i++)
   {
        scanf("%d", &array1[i]);
```

```
    }
    printf("\n Enter size of array 2: ");
    scanf("%d", &n);
    printf("\n Enter sorted elements of array 2: \n");
    for (i = 0; i < n; i++)
    {
            scanf("%d", &array2[i]);
    }
            do{
                    printf("\n\n--------Menu----------\n");
                    printf("1.mergesort\n");
                    printf("2.Exit\n");
                    printf("\nEnter your choice:\t");
                    scanf("%d",&choice);
                    switch(choice)
                    {
                            case 1:mergesort();break;
                            case 2:exit(0);break;
                            default:printf("\nInvalid choice:\n");break;
                    }
             }
            while(choice!=2);
            return 0;
}
void mergesort()
{
     i = 0;
    j = 0;
    k=0;
            while (i < m)
```

```
        {
           array3[k] = array1[i];
           i++;
           k++;
        }
        while (j < n)
        {
           array3[k] = array2[j];
           j++;
           k++;
        }
        for(i=0;i<m+n-1;i++)
        {
                for(j=0;j<m+n-1-i;j++)
                        if(array3[j]>array3[j+1])
                        {
                                temp=array3[j];
                                array3[j]=array3[j+1];
                                array3[j+1]=temp;
                        }
        }
         printf("\n After merging with sorting : \n");
   for (i = 0; i < m + n; i++)
   {
        printf("%d\t", array3[i]);
   }
}
```

// 1.(c). Write a program to perform the Matrix addition, Multiplication and Transpose Operation. //[Menu Driven]

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

```
#include<conio.h>
void display(int [][3]);
void func1();
void func2();
void func3();
int main()
{
  int c;
  clrscr();
  do{
  printf("\n Matrix Manipulation Functions  :");
  printf("\n1. Matrix Addition:");
  printf("\n2. Matrix Multiplication:");
  printf("\n3. Transpose Matrix:");
  printf("\n4. exit      :");
  printf("\n Enter Your Choice:");
  scanf("%d",&c);
  switch(c)
  {
     case 1:func1();break;
     case 2:func2();break;
     case 3:func3();break;
     case 4:exit(0);break;
     default:printf("\nInvalid Choice");
  }
    }
 while(c!=4);
 return 0;
}
void func1()
```

```
{
        int x[3][3],y[3][3],z[3][3];
        void getmatrix(int [][3]);
        void addition(int [][3],int [][3],int [][3]);
        clrscr();
        getmatrix(x);
        getmatrix(y);
        addition(x,y,z);
        printf("\n - : Matrix 1: - \n");
        display(x);
        printf("\n - : Matrix 2: - \n");
        display(y);
        printf("\n - : Matrix Addition (Result): - \n");
        display(z);
}
    void getmatrix(int t[][3])
    {
          int i,j;
          for(i=0;i<3;i++)
          {
             for(j=0;j<3;j++)
             {
                   printf("Enter element [%d][%d] : ",i,j);
                   scanf("%d",&t[i][j]);
             }
          }
    }
    void addition(int p[][3],int q[][3],int r[][3])
    {   int i,j;
          for(i=0;i<3;i++)
```

```
        {   for(j=0;j<3;j++)
                r[i][j]=p[i][j]+q[i][j];
        }
    }
void func2()
{
        int x[3][3],y[3][3],z[3][3];
        void getmatrix(int [][3]);
        void multiplication(int [][3],int [][3],int [][3]);
        clrscr();
        getmatrix(x);
        getmatrix(y);
        multiplication(x,y,z);
        printf("\n - : Matrix 1: - \n");
        display(x);
        printf("\n - : Matrix 2: - \n");
        display(y);
        printf("\n - : Matrix Multiplication (Result): - \n");
        display(z);
}
        void multiplication(int p[][3],int q[3][3],int r[3][3])
        {
                int i,j,k;
                for(i=0;i<3;i++)
                {
                        for(j=0;j<3;j++)
                        {
                                r[i][j]=0;
                                for(k=0;k<3;k++)
                                r[i][j]=r[i][j]+(p[i][j]*q[j][k]);
```

```
                        }
                    }
            }
void func3()
{
            int x[3][3],y[3][3];
            void getmatrix(int [][3]);
            void transpose(int [][3],int [][3]);
            clrscr();
            getmatrix(x);
            transpose(x,y);
            printf("\n - : Matrix 1: - \n");
            display(x);
            printf("\n - : Transpose Matrix : - \n");
            display(y);
}
            void transpose(int p[][3],int q[][3])
            {
              int i,j;
              for(i=0;i<3;i++)
              {
                 for(j=0;j<3;j++)
                        q[i][j]=p[j][i];
              }
      }
      void display(int m[][3])
      {
            int i,j;
            printf("\n");
            for(i=0;i<3;i++)
```

```
        {
          for(j=0;j<3;j++)
                printf("%d  ",m[i][j]);
          printf("\n");
        }
    }
```

//linked list

//2 .(a)Write a program to create a single linked list and display the node elements in reverse order

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int visited;
   int a;
   struct node *next;
};
void generate(struct node **);
void display(struct node *);
void linear(struct node *);
int main()
{
   struct node *head = NULL;
   clrscr ();
   generate(&head);
   printf("\nPrinting the list in linear order\n");
   linear(head);
   printf("\nPrinting the list in reverse order\n");
   display(head);
   getch ();
```

```
    return 0;
}
void display(struct node *head)
{
    struct node *temp = head, *prev = head;
    while (temp->visited == 0)
    {
            while (temp->next != NULL && temp->next->visited == 0)
            {
               temp = temp->next;
            }
            printf("%d  ", temp->a);
            temp->visited = 1;
            temp = head;
    }
}
void linear(struct node *head)
{
    while (head != NULL)
    {
            printf("%d  ", head->a);
            head = head->next;
    }
    printf("\n");
}
void generate(struct node **head)
{
    int num, i;
    struct node *temp;
    printf("Enter length of list: ");
```

```
   scanf("%d", &num);
   for (i = num; i > 0; i--)
   {
          temp = (struct node *)malloc(sizeof(struct node));
          temp->a = i;
          temp->visited = 0;
          if (*head == NULL)
          {
             *head = temp;
             (*head)->next = NULL;
          }
          else
          {
             temp->next = *head;
             *head = temp;
          }
   }
}
```

// 2 .(b)Write a program to search the elements in the linked list and display the same

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
  int data;
  struct node *next;
}
first, *nw;
int search(int);
```

```
void main()
{
        int no,i,item,pos;
        clrscr();
        first.next=NULL;
        nw=&first;
        printf("Enter The No of nodes, you want in linked list: ");
        scanf("%d",&no);
        for(i=0;i< no;i++)
        {
                nw->next=(struct node *)malloc(sizeof(struct node));
                printf("Enter element in node %d: ",i+1);
                scanf("%d",&nw->data);
                nw=nw->next;
        }
        nw->next=NULL;
        printf("Linked list is:\n");
        nw=&first;
        while(nw->next!=NULL)
        {
                printf("%d\t",nw->data);
                nw=nw->next;
        }
        printf("\nEnter item to be searched : ");
        scanf("%d",&item);
        pos=search(item);
        if(pos<=no)
                printf("Your item is at node=%d",pos);
        else
                printf("Sorry! your item is not in linked list.");
```

```
getch();
}
int search(int item)
{
        int count=1;
        nw=&first;
        while(nw->next!=NULL)
        {
                if(nw->data==item)
                break;
                else
                count++;
                nw=nw->next;
        }
return count;
}
```

// 2 .(c) Write a program to create double linked list and sort the elements in the linked list.

```
#include<stdio.h>
#include<conio.h>
struct Node
{
        int data;
        struct Node* prev, *next;

};
struct Node* getNode(int data)
{
        struct Node* newNode =
                (struct Node*)malloc(sizeof(struct Node));
```

```
	newNode->data = data;
	newNode->prev = newNode->next = NULL;
	return newNode;
}
void sortedInsert(struct Node** head_ref, struct Node* newNode)
{
	struct Node* current;
	if (*head_ref == NULL)
		*head_ref = newNode;
	else if ((*head_ref)->data >= newNode->data)
	{
		newNode->next = *head_ref;
		newNode->next->prev = newNode;
		*head_ref = newNode;
	}
	else
	{
		current = *head_ref;
		while (current->next != NULL &&
			current->next->data < newNode->data)
			current = current->next;
		newNode->next = current->next;
		if (current->next != NULL)
			newNode->next->prev = newNode;
		current->next = newNode;
		newNode->prev = current;
	}
}
void insertionSort(struct Node** head_ref)
{
```

```
	struct Node* sorted = NULL;
	struct Node* current = *head_ref;
	while (current != NULL)
	{
		struct Node* next = current->next;
		current->prev = current->next = NULL;
		sortedInsert(&sorted, current);
		current = next;
	}
	*head_ref = sorted;
}
void printList(struct Node* head)
{
	while (head != NULL)
	{
		printf(" ",head->data);
		head = head->next;
	}
}
void push(struct Node** head_ref, int new_data)
{
	struct Node* new_node =
		(struct Node*)malloc(sizeof(struct Node));
	new_node->data = new_data;
	new_node->next = (*head_ref);
	new_node->prev = NULL;
	if ((*head_ref) != NULL)
		(*head_ref)->prev = new_node;
	(*head_ref) = new_node;
}
```

```
void main()
{
        struct Node* head = NULL;
        clrscr();
        push(&head, 9);
        push(&head, 3);
        push(&head, 5);
        push(&head, 10);
        push(&head, 12);
        push(&head, 8);
        printf("Doubly Linked List Before Sortingn");
        printList(head);
        insertionSort(&head);
        printf("\nDoubly Linked List After Sortingn");
        printList(head);
        getch ();
}
```

//Stack.

//3 .(a). Write a program to implement the concept of Stack with Push, Pop, Display and Exit operations.

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

#define MAX 5   //Maximum number of elements that can be stored

int top=-1,stack[MAX];
void push();
void pop();
void display();
```

```
void main()
{
   int ch;

   while(1)    //infinite loop, will end when choice will be 4
   {
      printf("\n*** Stack Menu ***");
      printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");
      printf("\n\nEnter your choice(1-4):");
      scanf("%d",&ch);

      switch(ch)
      {
         case 1: push();
               break;
         case 2: pop();
               break;
         case 3: display();
               break;
         case 4: exit(0);

         default: printf("\nWrong Choice!!");
      }
   }
}

void push()
{
   int val;
```

```
   if(top==MAX-1)
   {
      printf("\nStack is full!!");
   }
   else
   {
      printf("\nEnter element to push:");
      scanf("%d",&val);
      top=top+1;
      stack[top]=val;
   }
}

void pop()
{
   if(top==-1)
   {
      printf("\nStack is empty!!");
   }
   else
   {
      printf("\nDeleted element is %d",stack[top]);
      top=top-1;
   }
}

void display()
{
   int i;
```

```
   if(top==-1)
   {
      printf("\nStack is empty!!");
   }
   else
   {
      printf("\nStack is...\n");
      for(i=top;i>=0;--i)
         printf("%d\n",stack[i]);
   }
}
```

// 3 .(b) Write a program to convert an infix expression to postfix and prefix conversion.

```
#define SIZE 50          /* Size of Stack */
#include<string.h>
#include <ctype.h>
char s[SIZE];
int top=-1;      /* Global declarations */
push(char elem)
{                  /* Function for PUSH operation */
   s[++top]=elem;
}
char pop()
{                 /* Function for POP operation */
   return(s[top--]);
}
int pr(char elem)
{             /* Function for precedence */
   switch(elem)
```

```
   {
   case '#': return 0;
   case ')': return 1;
   case '+':
   case '-': return 2;
   case '*':
   case '/': return 3;
   }
}
void main()
{                    /* Main Program */
   char infx[50],prfx[50],pofx[50],ch,elem;
   int i=0,k=0;
   clrscr();
   printf("\n\nRead the Infix Expression ? ");
   scanf("%s",infx);
   push('#');
   strrev(infx);
   while( (ch=infx[i++]) != '\0')
   {
          if( ch == ')') push(ch);
          else
             if(isalnum(ch)) prfx[k++]=ch;
             else
                    if( ch == '(')
                    {
                       while( s[top] != ')')
                              prfx[k++]=pop();
                       pop(); /* Remove ) */
                    }
```

```
                else
                {     /* Operator */
                   while( pr(s[top]) >= pr(ch) )
                           prfx[k++]=pop();
                   push(ch);
                }
   }
  /* while( s[top] != '#')
   {    /* Pop from stack till empty
           pofx[k++]=pop();
   pofx[k]='\0';         // Make pofx as valid string *
   printf("\n\nGiven Infix Expn: %s  Postfix Expn: %s\n",infx,pofx);
   }  */
   while( s[top] != '#')    /* Pop from stack till empty */
           prfx[k++]=pop();
   prfx[k]='\0';         /* Make prfx as valid string */
   strrev(infx);
   printf("\n\nGiven Infix Expn: %s  Postfix Expn: %s\n",infx,prfx);
   strrev(prfx);
   printf("\n\nGiven Infix Expn: %s  Prefix Expn: %s\n",infx,prfx);
   getch ();
}
```

// 3 .(c) Write a program to implement Tower of Hanoi problem.

```
// C program for Tower of Hanoi using Recursion
#include <stdio.h>
void towers(int, char, char, char);
void main()
{
   int num;
   clrscr ();
```

```
   printf("Enter the number of disks : ");
   scanf("%d", &num);
   printf("The sequence of moves involved in the Tower of Hanoi are :\n");
   towers(num, 'A', 'C', 'B');
   getch ();
   return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
   if (num == 1)
   {
          printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
          return;
   }
   towers(num - 1, frompeg, auxpeg, topeg);
   printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
   towers(num - 1, auxpeg, topeg, frompeg);
}
```

// 4 . (a) Write a program to implement the concept of Queue with Insert, Delete, Display and Exit operations.

```
// C Program to Implement a Queue using an Array
#include <stdio.h>
#include<conio.h>
#define MAX 50
int queue_array[MAX];
int rear = - 1;
int front = - 1;
void insert();
void delete();
void display();
```

```
void main()
{
   int choice;
   clrscr ();
   while (1)
   {
         printf("1.Insert element to queue \n");
         printf("2.Delete element from queue \n");
         printf("3.Display all elements of queue \n");
         printf("4.Quit \n");
         printf("Enter your choice : ");
         scanf("%d", &choice);
         switch (choice)
         {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
         } /*End of switch*/
   } /*End of while*/
   getch ();
```

```
} /*End of main()*/
void insert()
{
   int add_item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
   {
         if (front == - 1)
         /*If queue is initially empty */
         front = 0;
         printf("Inset the element in queue : ");
         scanf("%d", &add_item);
         rear = rear + 1;
         queue_array[rear] = add_item;
   }
} /*End of insert()*/

void delete()
{
   if (front == - 1 || front > rear)
   {
         printf("Queue Underflow \n");
         return ;
   }
   else
   {
         printf("Element deleted from queue is : %d\n", queue_array[front]);
         front = front + 1;
   }
```

```
} /*End of delete() */
void display()
{
   int i;
   if (front == - 1)
         printf("Queue is empty \n");
   else
   {
         printf("Queue is : \n");
         for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
         printf("\n");
   }
} /*End of display() */
```

// 4 .(b) Write a program to implement the concept of Circular Queue.

```
#include<stdio.h>
#define max 3
int q[10],front=0,rear=-1;
void main()
{
   int ch;
   void insert();
   void delet();
   void display();
   clrscr();
   printf("\nCircular Queue operations\n");
   printf("1.insert\n2.delete\n3.display\n4.exit\n");
   while(1)
   {
      printf("Enter your choice:");
```

```
        scanf("%d",&ch);
        switch(ch)
        {
        case 1: insert();
           break;
        case 2: delet();
           break;
        case 3:display();
           break;
        case 4:exit();
        default:printf("Invalid option\n");
        }
    }
}

void insert()
{
    int x;
    if((front==0&&rear==max-1)||(front>0&&rear==front-1))
        printf("Queue is overflow\n");
    else
    {
        printf("Enter element to be insert:");
        scanf("%d",&x);
        if(rear==max-1&&front>0)
        {
            rear=0;
            q[rear]=x;
        }
        else
```

```
      {
         if((front==0&&rear==-1)||(rear!=front-1))
            q[++rear]=x;
      }
   }
}
void  delet()
{
   int a;
   if((front==0)&&(rear==-1))
   {
      printf("Queue is underflow\n");
      getch();
      exit();
   }
   if(front==rear)
   {
      a=q[front];
      rear=-1;
      front=0;
   }
   else
      if(front==max-1)
      {
         a=q[front];
         front=0;
      }
      else a=q[front++];
      printf("Deleted element is:%d\n",a);
}
```

```
void display()
{
   int i,j;
   if(front==0&&rear==-1)
   {
      printf("Queue is underflow\n");
      getch();
      exit();
   }
   if(front>rear)
   {
      for(i=0;i<=rear;i++)
         printf("\t%d",q[i]);
      for(j=front;j<=max-1;j++)
         printf("\t%d",q[j]);
      printf("\nrear is at %d\n",q[rear]);
      printf("\nfront is at %d\n",q[front]);
   }
   else
   {
      for(i=front;i<=rear;i++)
      {
         printf("\t%d",q[i]);
      }
      printf("\nrear is at %d\n",q[rear]);
      printf("\nfront is at %d\n",q[front]);
   }
   printf("\n");
getch();
```

```
}
```

// 4 .(c) Write a program to implement the concept of Deque.

```
#define MAX 100
#include<stdio.h>
#include<conio.h>
void insert(int);
int delStart();
int delEnd();
int queue[MAX];
int rear =0, front =0;
void display();
void insertEnd(int);
void insertStart(int);
void main()
{
   char ch , a='y';
   int choice, c, token;
   clrscr ();
   printf("Enter your choice for which deque operation you want to perform operation \n");
    do
    {
          printf("\n1.Input-restricted deque \n");
          printf("2.output-restricted deque \n");
          printf("\nEnter your choice for the operation : ");
          scanf("%d",&c);
          switch(c)
          {
              case 1:
                    printf("\nDo operation in Input-Restricted c deque\n");
                    printf("1.Insert");
                    printf("\n2.Delete from end");
                    printf("\n3.Delete from begning");
                    printf("\n4.show or display");
                    do
                   {
                   printf("\nEnter your choice for the operation in c deque: ");
                   scanf("%d",&choice);
                   switch(choice)
                   {
                      case 1: insertEnd(token);
                      display();
                      break;
                      case 2: token=delEnd();
                      printf("\nThe token deleted is %d",token);
                      display();
                      break;
                      case 3: token=delStart();
                      printf("\nThe token deleted is %d",token);
                      display();
                      break;
```

```
                    case 4: display();
                    break;
                    default:printf("Wrong choice");
                    break;
                    }
                    printf("\nDo you want to continue(y/n) to do operation in input-restricted c
deque: ");
                    ch=getch();
                    }
                    while(ch=='y'||ch=='Y');
                    getch();
                    break;

               case 2 :
                    printf("\nDo operation in Output-Restricted c deque\n");
                    printf("1.Insert at the End");
                    printf("\n2.Insert at the begning");
                    printf("\n3.Delete the element");
                    printf("\n4.show or display");
                    do
                    {
                    printf("\nEnter your choice for the operation: ");
                    scanf("%d",&choice);
                    switch(choice)
                       {
                       case 1: insertEnd(token);
                       display();
                       break;
                       case 2: insertStart(token);
                       display();
                       break;
                       case 3: token=delStart();
                       printf("\nThe token deleted is %d",token);
                       display();
                       break;
                       case 4: display();
                       break;
                       default:printf("Wrong choice");
                       break;
                       }
                    printf("\nDo you want to continue(y/n):");
                    ch=getch();
                    }
                    while(ch=='y'||ch=='Y');
                    getch();
                    break ;
          }
    printf("\nDo you want to continue(y/n):");
                    ch=getch();
                    }
                    while(ch=='y'||ch=='Y');
```

```
                    getch();
}
void display()
{
    int i;
    printf("\nThe queue elements are:");
    for(i=rear;i<front;i++)
    {
            printf("%d  ",queue[i]);
    }
}
void insertEnd(int token)
{
    char a;
    if(front==MAX/2)
     {
            printf("\nQueue full\nyou cant enter more elements at the end of c queue");
            return;
     }
     do
     {
            printf("\nEnter the token to be inserted:");
            scanf("%d",&token);
            queue[front]=token;
            front=front+1;
            printf("do you want to continue insertion Y/N");
            a=getch();
     }
     while(a=='y');
}
void insertStart(int token)
{
    char a;
    if(front==MAX/2)
     {
            printf("\nQueue full\nyou cant enter more elements at the start of queue");
            return;
     }
     do
     {
            printf("\nEnter the token to be inserted:");
            scanf("%d",&token);
            rear=rear-1;
            queue[rear]=token;
            printf("do you want to continue insertion Y/N");
            a=getch();
     }
     while(a=='y');
}
int delEnd()
{
```

```
    int t;
    if(front==rear)
    {
            printf("\nQueue empty");
            return 0;
    }
    front=front-1;
    t=queue[front+1];
    return t;
}
int delStart()
{
    int t;
    if(front==rear)
    {
            printf("\nQueue empty");
            return 0;
    }
    rear=rear+1;
    t=queue[rear-1];
    return t;
}
```

// 5 (a) Write a program to implement bubble sort.

```
// C Program to sort an array using Bubble Sort technique
#include <stdio.h>
#include<conio.h>
void swap();
void bubblesort(int arr[], int size)
{
   int i, j;
   for (i = 0;  i < size; i++)
   {
          for (j = 0; j < size - i; j++)
          {
             if (arr[j] > arr[j+1])
                    swap(&arr[j], &arr[j+1]);
          }
   }
}
void swap(int *a, int *b)
{
   int temp;
   temp = *a;
   *a = *b;
   *b = temp;
}
void main()
{
   int array[100], i, size;
   clrscr();
   printf("How many numbers you want to sort:  ");
   scanf("%d", &size);
```

```
    printf("\nEnter %d numbers : ", size);
    for (i = 0; i < size; i++)
            scanf("%d", &array[i]);
    bubblesort(array, size);
    printf("\nSorted array is ");
    for (i = 0; i <size; i++)
            printf(" %d ", array[i]);
    getch();
}
```

// 5 .(b) Write a program to implement selection sort.

```
#include <stdio.h>
#include<conio.h>
void main()
{
  int array[100], n, c, d, position, swap;
  clrscr();
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for ( c = 0 ; c < n ; c++ )
    scanf("%d", &array[c]);
  for ( c = 0 ; c < ( n - 1 ) ; c++ )
  {
    position = c;
    for ( d = c + 1 ; d < n ; d++ )
    {
          if ( array[position] > array[d] )
            position = d;
    }
    if ( position != c )
    {
          swap = array[c];
          array[c] = array[position];
          array[position] = swap;
    }
  }
  printf("Sorted list in ascending order:\n");
  for ( c = 0 ; c < n ; c++ )
    printf("%d\n", array[c]);
  getch ();
}
```

// 5 .(c) Write a program to implement insertion sort.

```
/* insertion sort ascending order */
#include <stdio.h>
#include<conio.h>
void main()
{
 int n, array[1000], c, d, t;
 clrscr();
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (c = 0; c < n; c++) {
```

```
    scanf("%d", &array[c]);
  }
  for (c = 1 ; c <= n - 1; c++) {
    d = c;
    while ( d > 0 && array[d] < array[d-1]) {
      t          = array[d];
      array[d]   = array[d-1];
      array[d-1] = t;
      d--;
    }
  }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
  }
  getch ();
}
```

//6.(a) Write a program to implement merge sort.

```
#include <stdio.h>
#define max 10
#include<conio.h>
int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];
void merging(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 sort(int low, int high)
{
   int mid;
   if(low < high)
   {
      mid = (low + high) / 2;
      sort(low, mid);
      sort(mid+1, high);
      merging(low, mid, high);
   }
   else
   {
      return;
```

```
   }
}
void main()
{
   int i;
   clrscr();
   printf("List before sorting\n");
   for(i = 0; i <= max; i++)
      printf("%d ", a[i]);
   sort(0, max);
   printf("\nList after sorting\n");
   for(i = 0; i <= max; i++)
      printf("%d ", a[i]);
   getch ();
}
```

// 6 .(b) Write a program to search the element using sequential search.

```
#include<stdio.h>
#include<conio.h>
int search(int *,int,int);
void main()
{
 int num[100],loc=-1,i,n,ele;
 clrscr();
 printf("enter the number of elements to be inserted:");
 scanf("%d",&n);
 printf("\nenter the elements :");
 for(i=0;i<n;i++)
 scanf("%d",&num[i]);
 printf("\n enter the element to be searched:");
 scanf("%d",&ele);
 loc=search(num,ele,n);
 if(loc<0)
 printf("\n elements not found :");
 else
 printf("elements found at %d position",loc+1);
 getch();
 }
 int search(int num[],int item,int n)
 {
  int i;
  for(i=0;i<n;i++)
  {
   if(num[i]==item)
   return i;
   }
  return -1;
  }
```

// 6 .(c) Write a program to search the element using binary search.

```
#include <stdio.h>
#include<conio.h>
void main()
{
   int c, first, last, middle, n, search, array[100];
```

```
  clrscr ();
  printf("Enter number of elements\n");
  scanf("%d",&n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d",&array[c]);
  printf("Enter value to find\n");
  scanf("%d", &search);
  first = 0;
  last = n - 1;
  middle = (first+last)/2;
  while (first <= last)
  {
    if (array[middle] < search)
          first = middle + 1;
    else if (array[middle] == search)
    {
          printf("%d found at location %d.\n", search, middle+1);
          break;
    }
    else
          last = middle - 1;
    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d is not present in the list.\n", search);
  getch ();
}
```

//7.(a) Write a program to create the tree and display the elements.

```
#include <stdio.h>
#include <malloc.h>
#include <stdio.h>
#include <conio.h>
typedef struct TREE {
          int data;
          struct TREE *left;
          struct TREE *right;
}
TREE;
int main() {
          int data,depth;
          TREE *tree =NULL;
          TREE *InsertTree(int data,TREE *p);
          TREE *PrintTreeTriangle(TREE *tree, int level);
          int TreeDepth(TREE *tree,int *depth,int level);
          clrscr();
          while(1) {
                    printf("\nKey to insert|");
                    scanf("%d", &data);
                    if (data==0)
                              break;
                    tree =InsertTree(data,tree);
```

```
                printf("\n Tree Display;\n");
                PrintTreeTriangle(tree,1);
                depth=0;
                TreeDepth(tree,&depth,0);
                printf("\nTree Depth =%d\n",depth);
        }
        return(0);
}
TREE *InsertTree(int data,TREE *p) {
        if(!p) {
                p=(TREE*)malloc(sizeof(TREE));
                p->data=data;
                p->left=NULL;
                p->right=NULL;
                return(p);
        }
        if(data < p->data)
                p->left=InsertTree(data,p->left); else
                if(data > p->data)
                p->right=InsertTree(data,p->right);
        return(p);
}
TREE *PrintTreeTriangle(TREE *tree, int level) {
        int i;
        if(tree) {
                PrintTreeTriangle(tree->right,level+1);
                printf("\n\n");
                for (i=0;i<level;i++)
                                printf("       ");
                printf("%d",tree->data);
                PrintTreeTriangle(tree->left,level+1);
        }
        return(NULL);
}
int TreeDepth(TREE *tree,int *depth,int level) {
        if(tree) {
                if (level>*depth)
                    *depth=level;
                TreeDepth(tree->left,depth,level+1);
                TreeDepth(tree->right,depth,level+1);
        }
        return(0);
}
```

// 7 .(b) (c) . Write a program to construct the binary tree for inorder, postorder and preorder traversal of tree .

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
struct node
{
    int data;
    struct node* left;
    struct node* right;
```

```
};
struct node* newNode(int data)
{
    struct node* node = (struct node*)
    malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
void printPostorder(struct node* node)
{
    if (node == NULL)
          return;
    printPostorder(node->left);
           printPostorder(node->right);
    printf("%d ", node->data);
}
void printInorder(struct node* node)
{
    if (node == NULL)
           return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}
void printPreorder(struct node* node)
{
    if (node == NULL)
           return;
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}
void main()
{
    struct node *root  = newNode(1);
    root->left            = newNode(2);
    root->right          = newNode(3);
    root->left->left     = newNode(4);
    root->left->right   = newNode(5);
    clrscr ();
    printf("\npreorder traversal of binary tree is. \n");
    printPreorder(root);
    printf("\nInorder traversal of binary tree is. \n");
    printInorder(root);
    printf("\nPostorder traversal of binary tree is. \n");
    printPostorder(root);
    getch ();
}
```