SYIT DATA Structure Journal 2021

Data Structures (University of Mumbai)

SYIT DATA STRUCTURE JOURNAL 2021-2022

NAME: SHOBHA CHOUHAN

Roll NO. 37

SUBJECT: DATA STRUCTURE

INDEX

SR NO.	PRACTICAL	DATE	SIGN
1A	Write a program to store the elements in 1-D array and perform the operations like searching, sorting and reversing the elements. [Menu Driven]		
1B	Read the two arrays from the user and merge them and display the elements in sorted order.[Menu Driven]		
1C	Write a program to perform the Matrix addition, Multiplication and Transpose Operation. [Menu Driven]		
2A	Write a program to create a single linked list and display the node elements in reverse order.		
2B	Write a program to search the elements in the linked list and display the same		
2C	Write a program to create double linked list and sort the elements in the linked list.		
3A	Write a program to implement the concept of Stack with Push, Pop, Display and Exit operations.		
3B	Write a program to convert an infix expression to postfix and prefix conversion.		
3C	Write a program to implement Tower of Hanoi problem.		
4A	Write a program to implement the concept of Queue with Insert, Delete, Display and Exit operations.		
4B	Write a program to implement the concept of Circular Queue		
4C	Write a program to implement the concept of Deque.		
5A	Write a program to implement bubble sort.		
5B	Write a program to implement selection sort.		
5C	Write a program to implement insertion sort.		
6A	Write a program to implement merge sort.		
6B	Write a program to search the element using sequential search		
6C	Write a program to search the element using binary search.		
7A	Write a program to create the tree and display the elements.		
7B	Write a program to construct the binary tree.		

7C	**Write a program for inorder, postorder and preorder traversal of tree**		
8A	**Write a program to insert the element into maximum heap.**		
8B	**Write a program to insert the element into minimum heap.**		
9A	**Write a program to implement the collision technique.**		
9B	**Write a program to implement the concept of linear probing.**		
10A	**Write a program to generate the adjacency matrix.**		
10B	**Write a program for shortest path diagram.**		

Implement the following:

PRACTICAL 1A

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

INPUT:

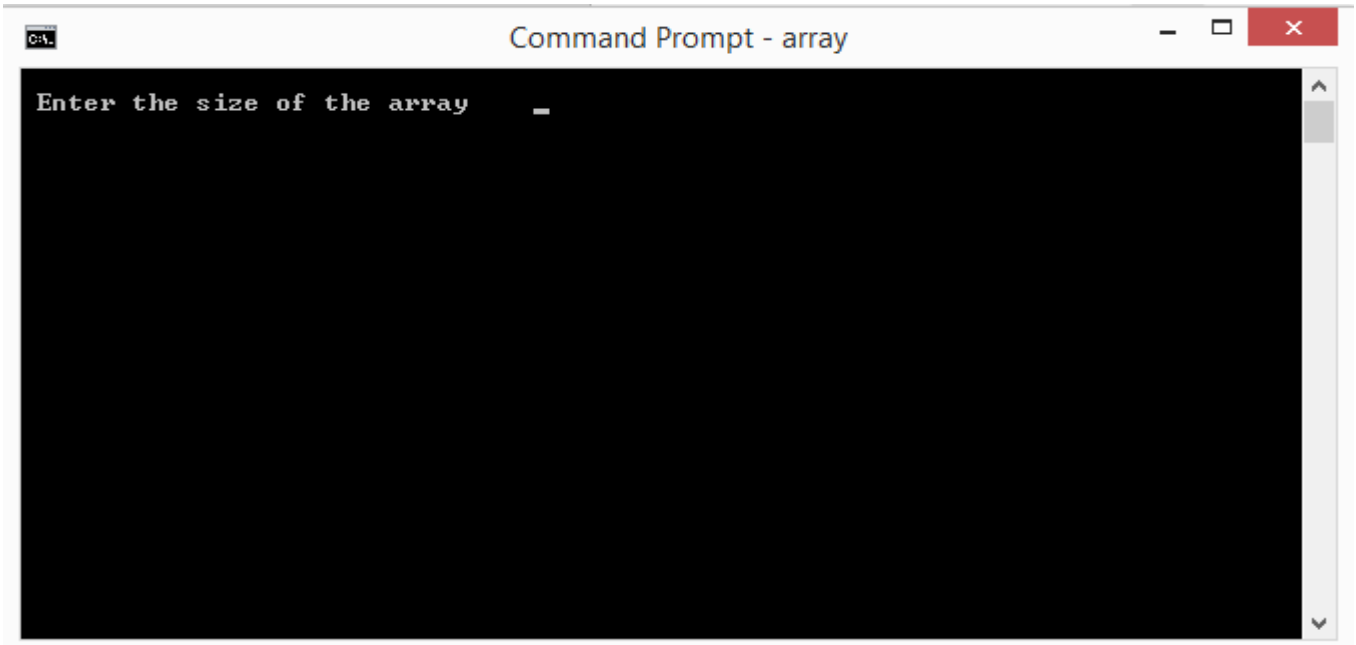
```
#include<stdio.h>
#include<conio.h>
int main()
{
int a[100],n, i ,I,j,x,ele,ch,flag=0;
clrscr();
printf("\n Enter the size of the array \t");
scanf("%d",&n);
printf("\n Enter the elements of the array: \n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
do
{
clrscr();
printf("\n 1-Searching \n 2-Sorting the array \n 3-Reverse the elements \n 4-Exit");
printf("\n Enter your choice\t");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the element to be searched \t");
scanf("%d",&ele);
for(i=0;i<n;i++)
```

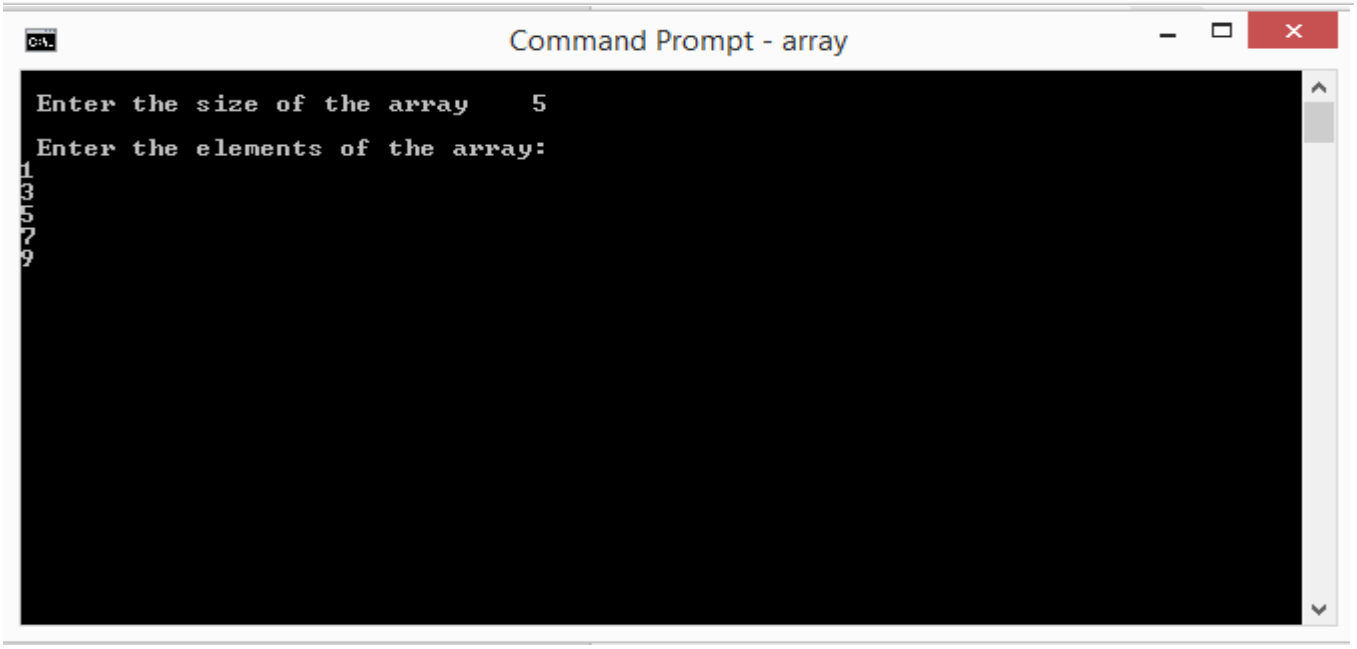
```
{
if(a[i]==ele)
{ x=
I;
flag=1;
break;
}
}
if(flag==0)
{
printf("\n Element not found!");
}
else
{
printf("\n Element found at position %d",x+1);
flag=0;
}
getch();
break;
case 2:
for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(a[j]>a[j+1])
{
x=a[j];
a[j]=a[j+1];
a[j+1]=x;
}
}
}
```

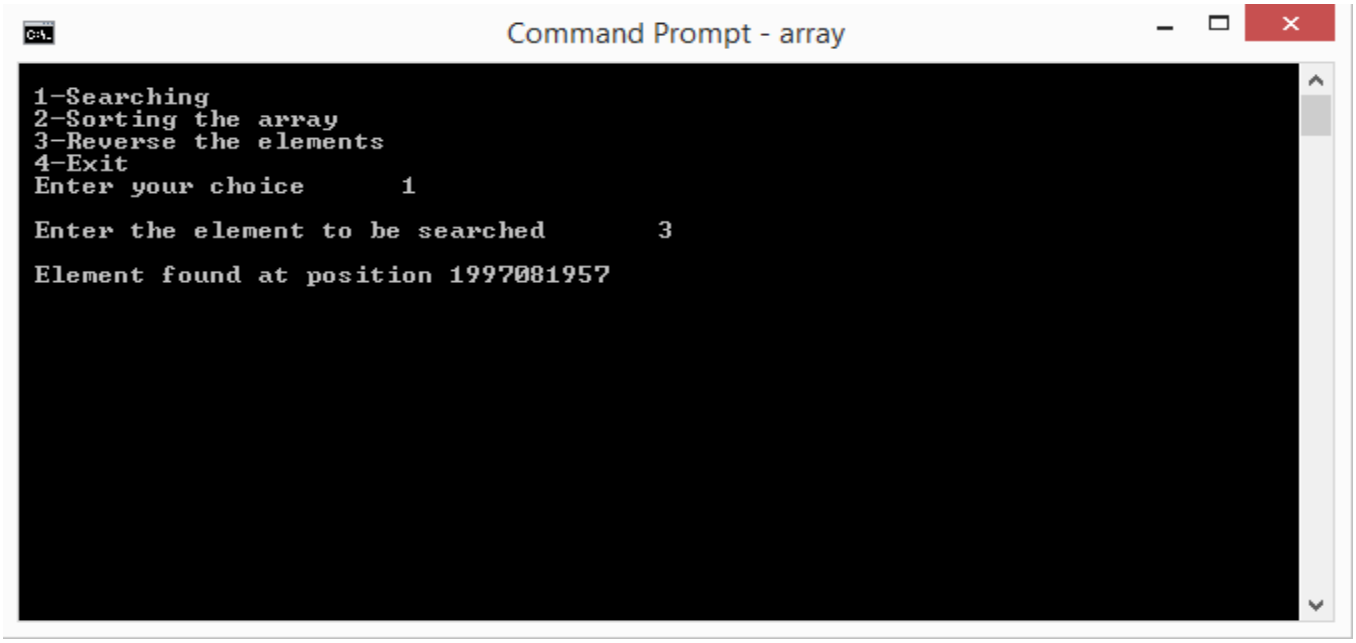
```
printf("\n After sorting\n");
for(i=0;i<n;i++)
{
printf("%d \t ",a[i]);
}
getch();
break;
case 3:
x=n/2;
j=n-1;
for(i=0;i<x;i++)
{
ele=a[i];
a[i]=a[j];
a[j]=ele;
j--;
}
printf("\n Reversed array is \n");
for(i=0;i<n;i++)
{
printf("%d \t ",a[i]);
}
getch();
break;
case 4:
exit(0);
break;
default:
printf("\n INVALID INPUT!");
}
}while(ch!=4);getch();
return 0;
```

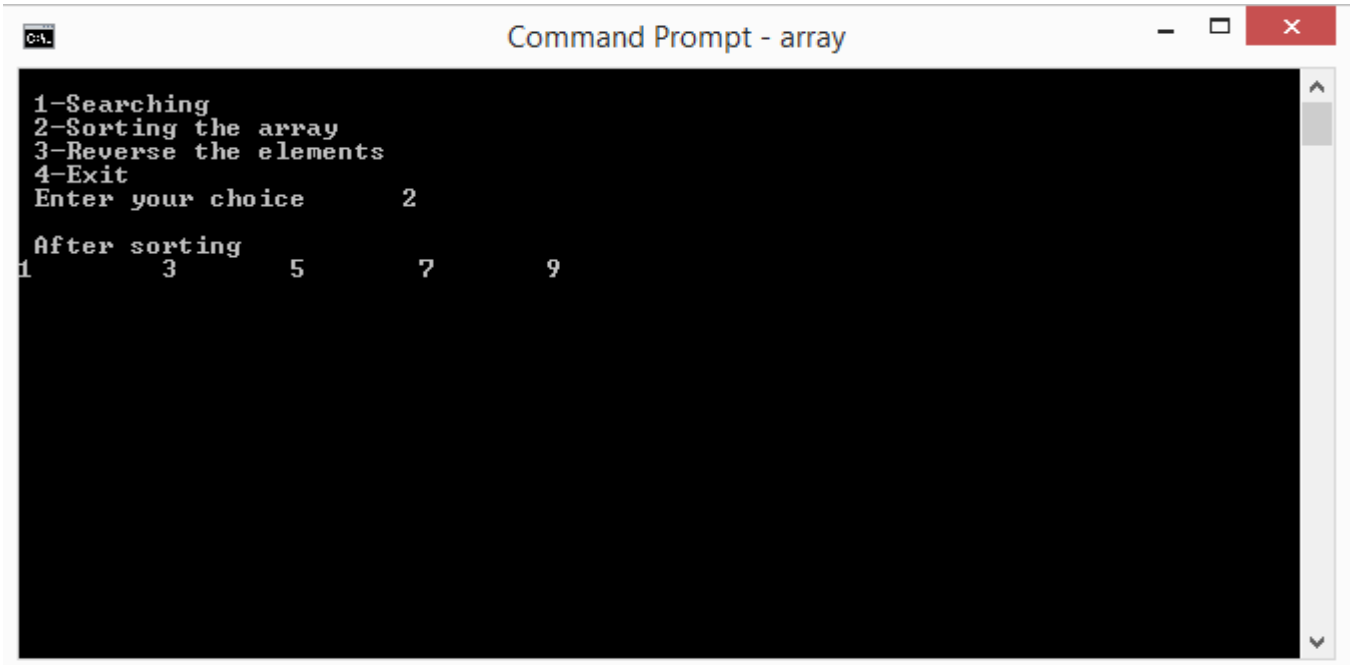
}

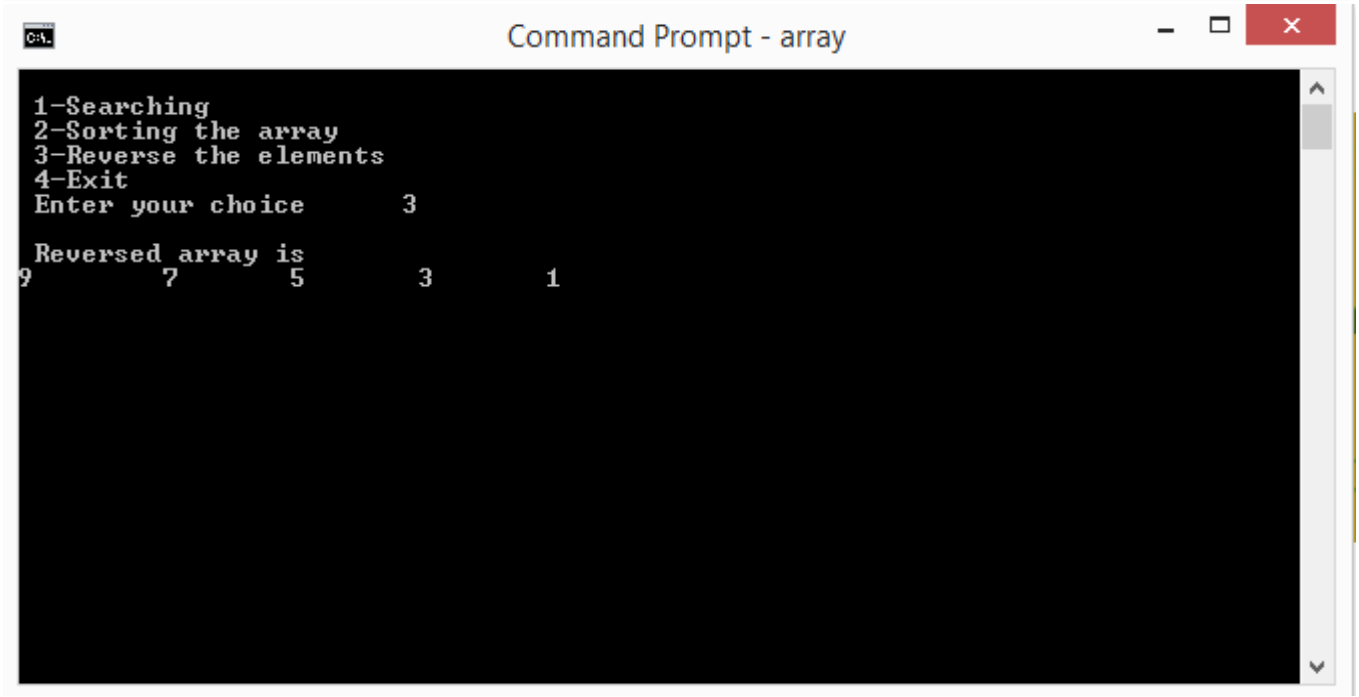
OUTPUT:











PRACTICAL 1B

Read the two arrays from the user and merge them and display the elements in sorted order.[Menu Driven]

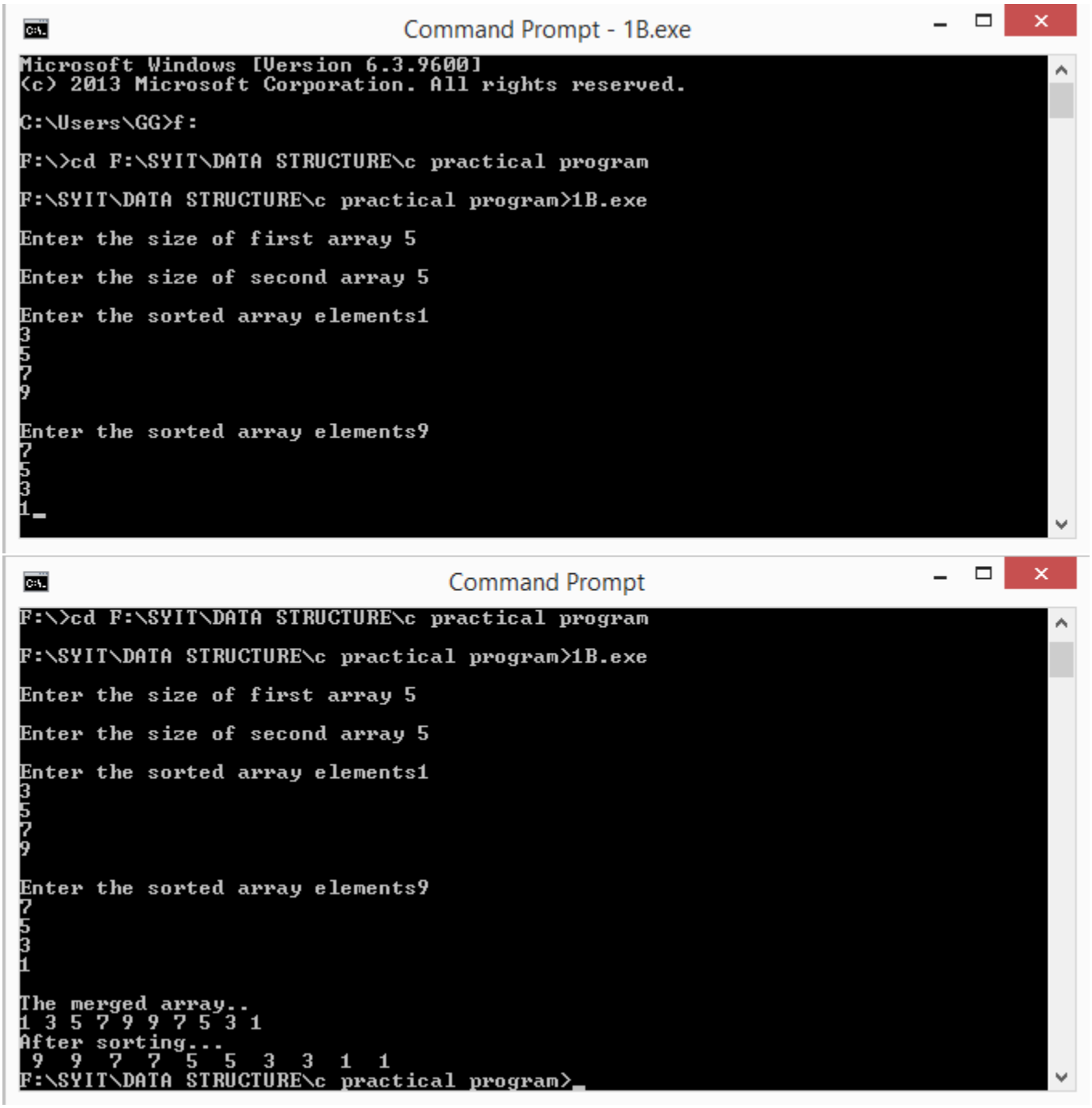
INPUT:

```
#include<stdio.h>
#include<conio.h>
intmain()
{
int arr1[20],arr2[20],arr3[40];
inti,j,k,size1,size2,temp;
printf("Enter the array size of array 1 :");
scanf("%d",&size1);
printf("Enter the element in arra 1 :\n");
for(i=0;i<size1;i++)
{
scanf("%d",&arr1[i]);
}
printf("Enter the size of array 2 : ");
scanf("%d",&size2);
printf("Enter the element in array2 : \n");
for(j=0;j<size2;j++)
{
scanf("%d",&arr2[j]);
}
for(i=0;i<size1;i++)
{
for(j=0;j<size1-i-1;j++)
{
if(arr1[j]>arr1[j+1])
{
temp=arr1[j];
```

```
arr1[j]=arr1[j+1];
arr1[j+1]=temp;
}
}
}
for(i=0;i<size2;i++)
{
for(j=0;j<size2-i-1;j++)
{
if(arr2[j]>arr2[j+1])
{
temp=arr2[j];
arr2[j]=arr2[j+1];
arr2[j+1]=temp;
}
}
}
/* printf("Sorting array 1\n");
for(i=0;i<size1;i++)
{
printf("%d \n",arr1[i]);
}
printf("Sorting array 2\n");
for(i=0;i<size2;i++)
{
printf("%d \n",arr2[i]);
}*/
i=0;
j=0;
k=0;
while(i<size1 && j<size2)
{
```

```
if(arr1[i]<arr2[j])
{
arr3[k]=arr1[i];
i++;
k++;
}
else
{
arr3[k]=arr2[j];
j++;
k++;
}
}
while(i<size1)
{
arr3[k]=arr1[i];
i++;
k++;
}
while(j<size2)
{
arr3[k]=arr2[j];
j++;
k++;
}
printf("merged array \n");
for(k=0;k<size1+size2;k++)
printf("%d \n",arr3[k]);
getch();
return 0;
}
```

OUTPUT:



PRACTICAL 1C

Write a program to perform the Matrix addition, Multiplication and Transpose Operation. [Menu Driven]

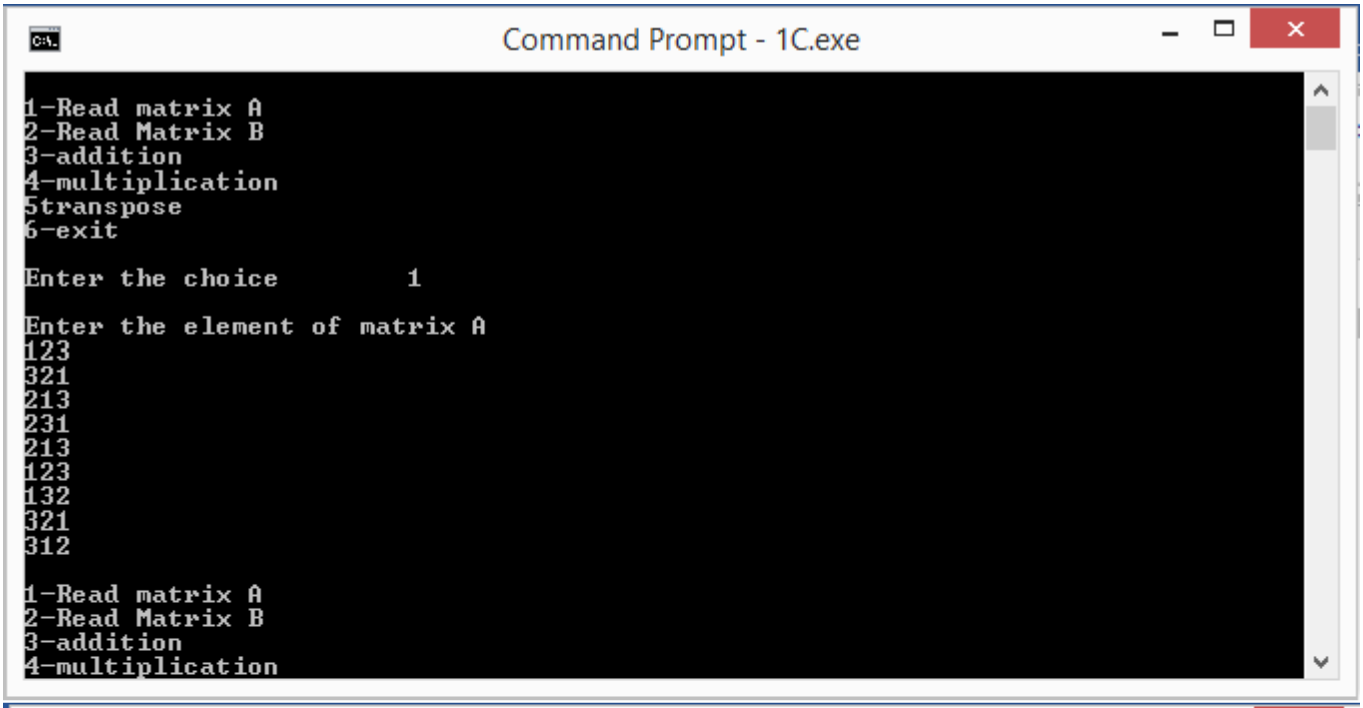
INPUT:

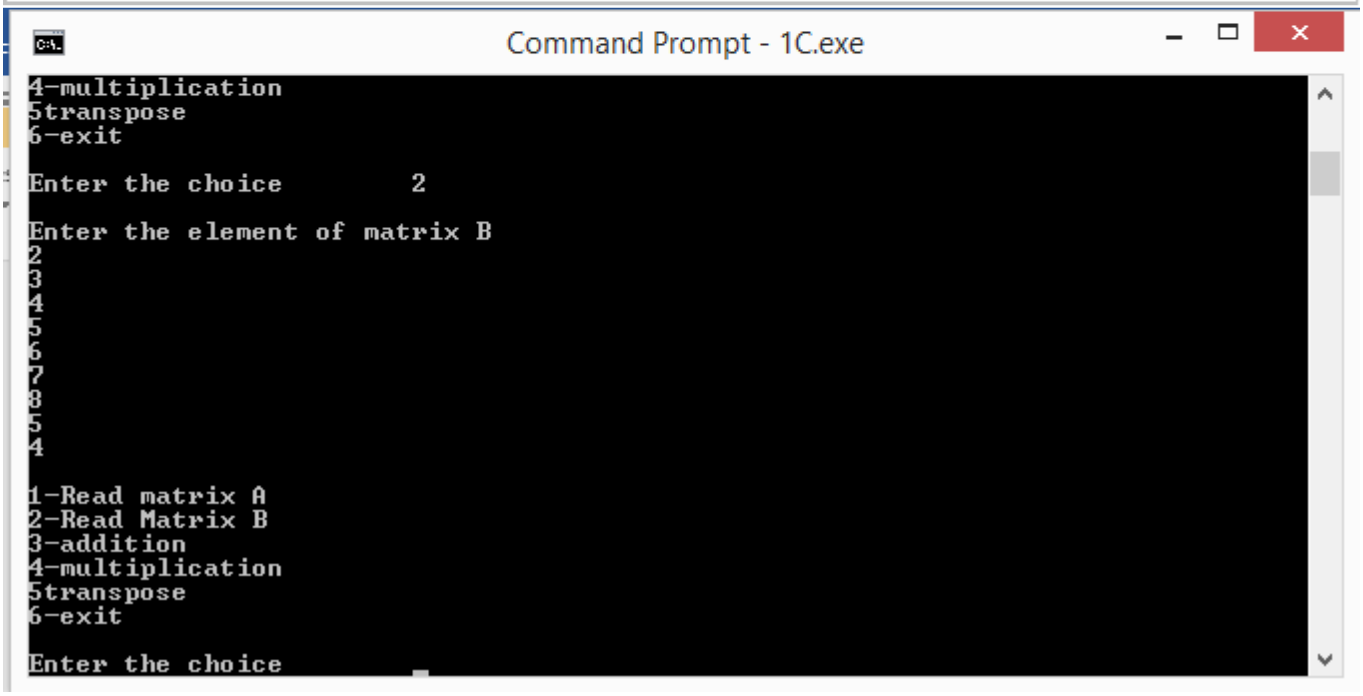
```
#include<stdio.h>
 #include<conio.h>
 int main()
{
 int a[3][3],b[3][3],c[3][3],i,j,k,ch;
 clrscr();
do
{ printf("\n1-Read matrix A\n2-Read Matrix B\n3-addition \n4-multiplication\n5transpose \n6-exit \n");
printf("\nEnter the choice \t");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1:
 {     printf("\nEnter the element of matrix A\n");
    for(i=0;i<3;i++)
  {      for(j=0;j<3;j++)
    {       scanf("%d",&a[i][j]);      }
   }
  break;
 }
   case 2:
  {     printf("\nEnter the element of matrix B\n");
     for(i=0;i<3;i++)
{      for(j=0;j<3;j++)
  {       scanf("%d",&b[i][j]);      }
  }
 break;     }
```

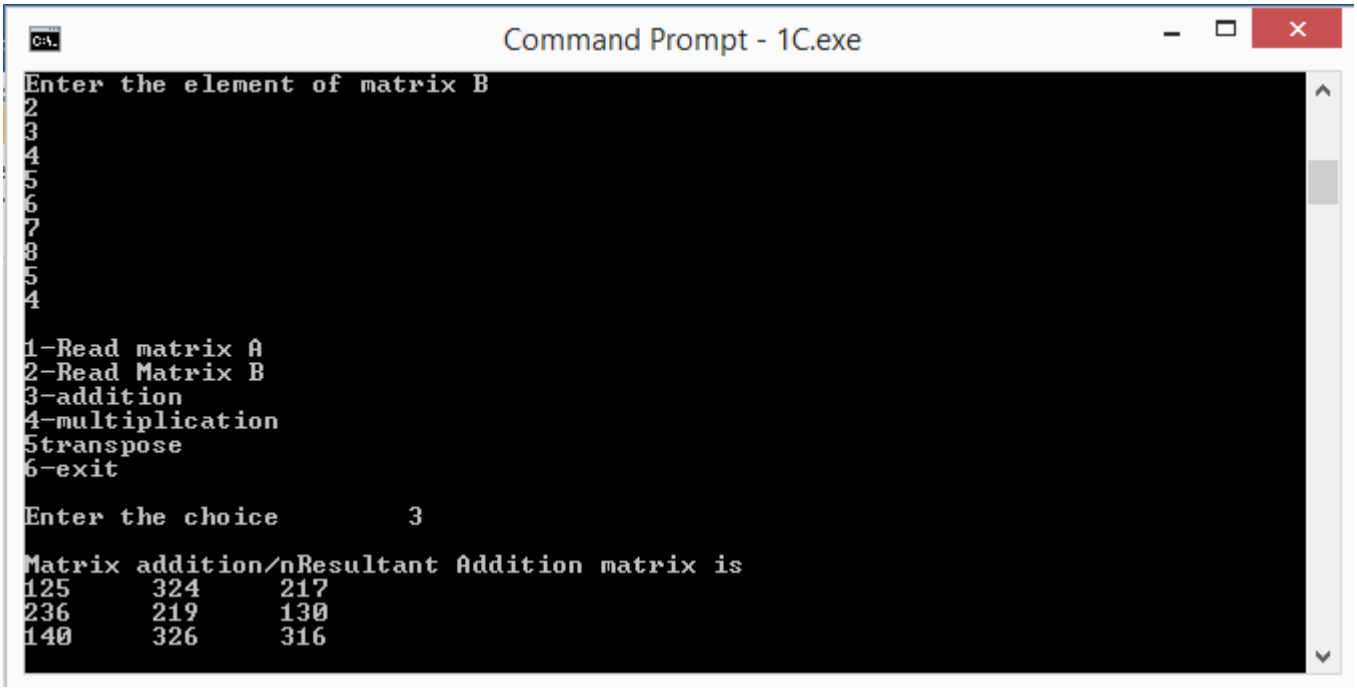
```
 case 3:
{     printf("\nMatrix addition");
  for(i=0;i<3;i++)
{      for(j=0;j<3;j++)
{       c[i][j]=a[i][j]+b[i][j];      }
   }
 printf("/nResultant Addition matrix is\n");
 for(i=0;i<3;i++)
 {          for(j=0;j<3;j++)
   {       printf("%d\t",c[i][j]);
     }
    printf("\n");
  }
getch();
 break;
}
 case 4:
 {     for(i=0;i<3;i++)
  {      for(j=0;j<3;j++)
  {       c[i][j]=0;      }
  }
  for(i=0;i<3;i++)
  {      for(j=0;j<3;j++)
    {       for(k=0;k<3;k++)
   {        c[i][j]=c[i][j]+(a[i][k]*b[k][j]);       }
   }
 }
 for(i=0;i<3;i++)
 {      for(j=0;j<3;j++)
  {       printf("%d\t",c[i][j]);
    }      printf("\n");
   }
```

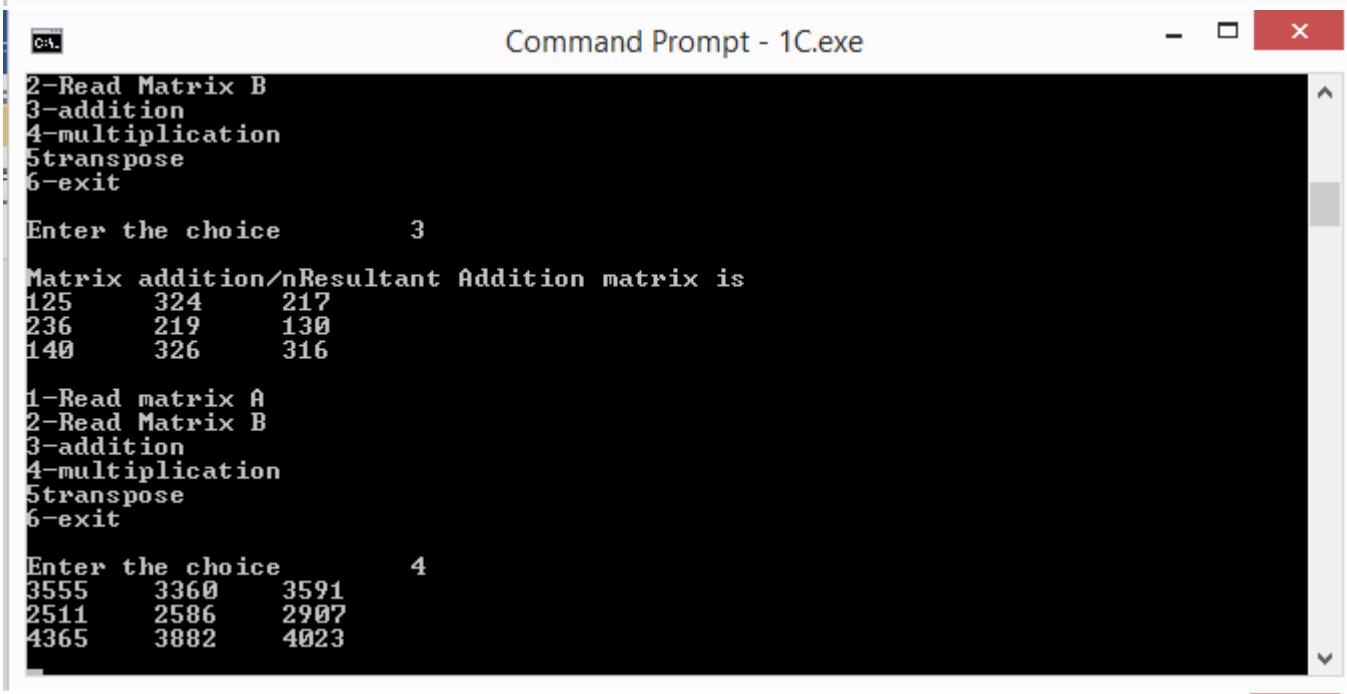
```
getch();
  break;
 }   case 5:
 {    printf("\nTranspose matrix\n");
  for(i=0;i<3;i++)
 {     for(j=0;j<3;j++)
 {      printf("%d\t",a[j][i]);
 }     printf("\n");
}
getch();
  break;
 }
case 6:
 {
  exit(0);
 break;
 }
default:
 {
  printf("\nInvalid input");
 }
 }
} while(ch!=6);
    getch();
}
```

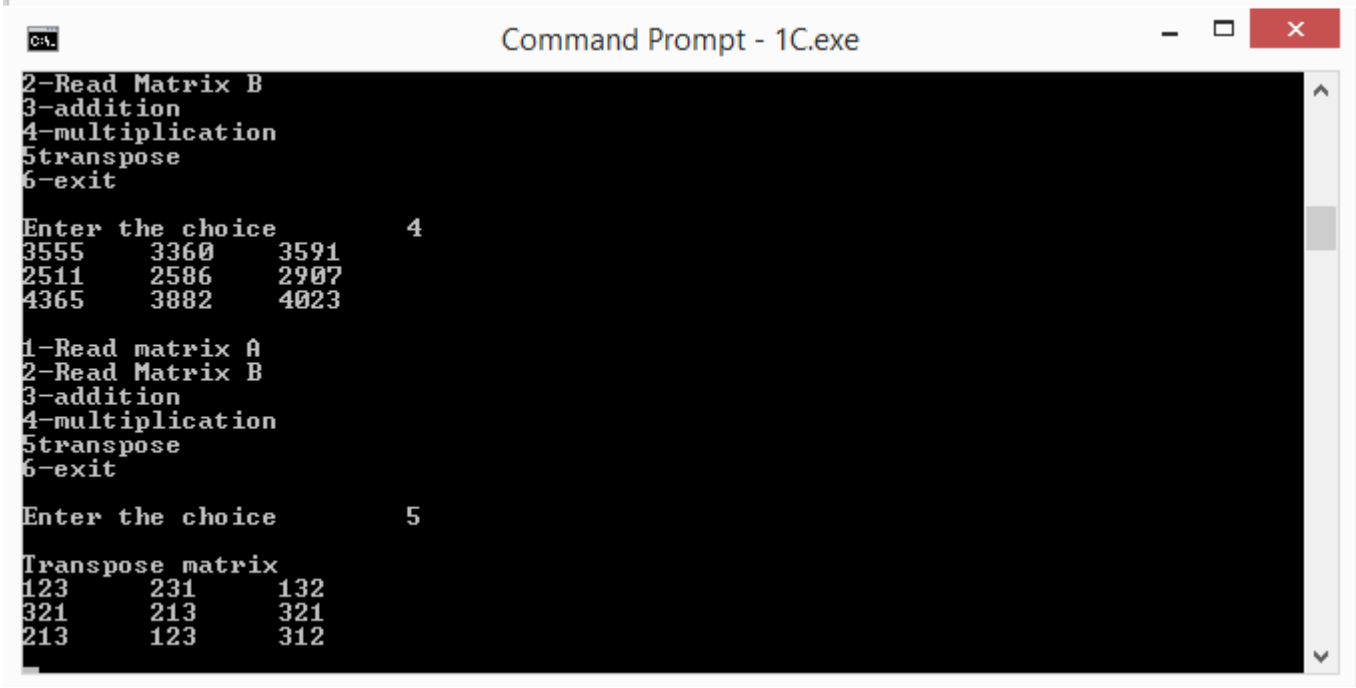
OUTPUT:











Implement the following for Linked List:

PRACTICAL 2A

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

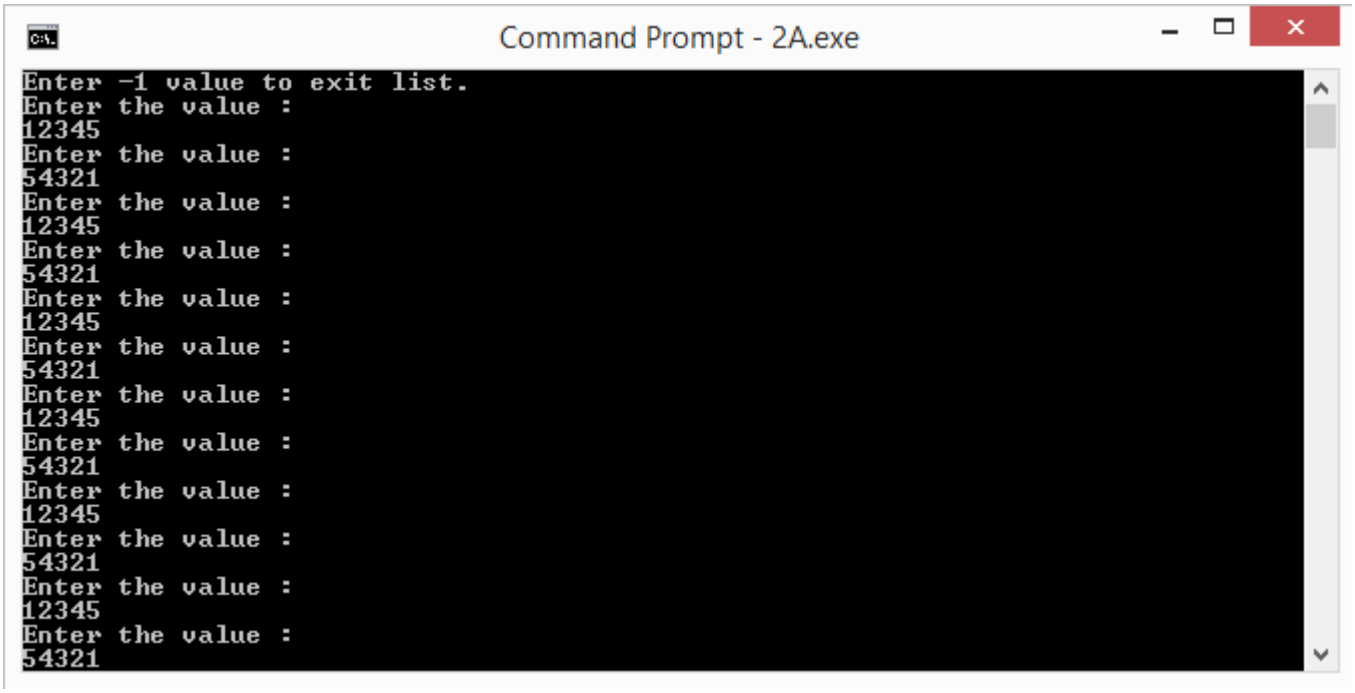
INPUT:

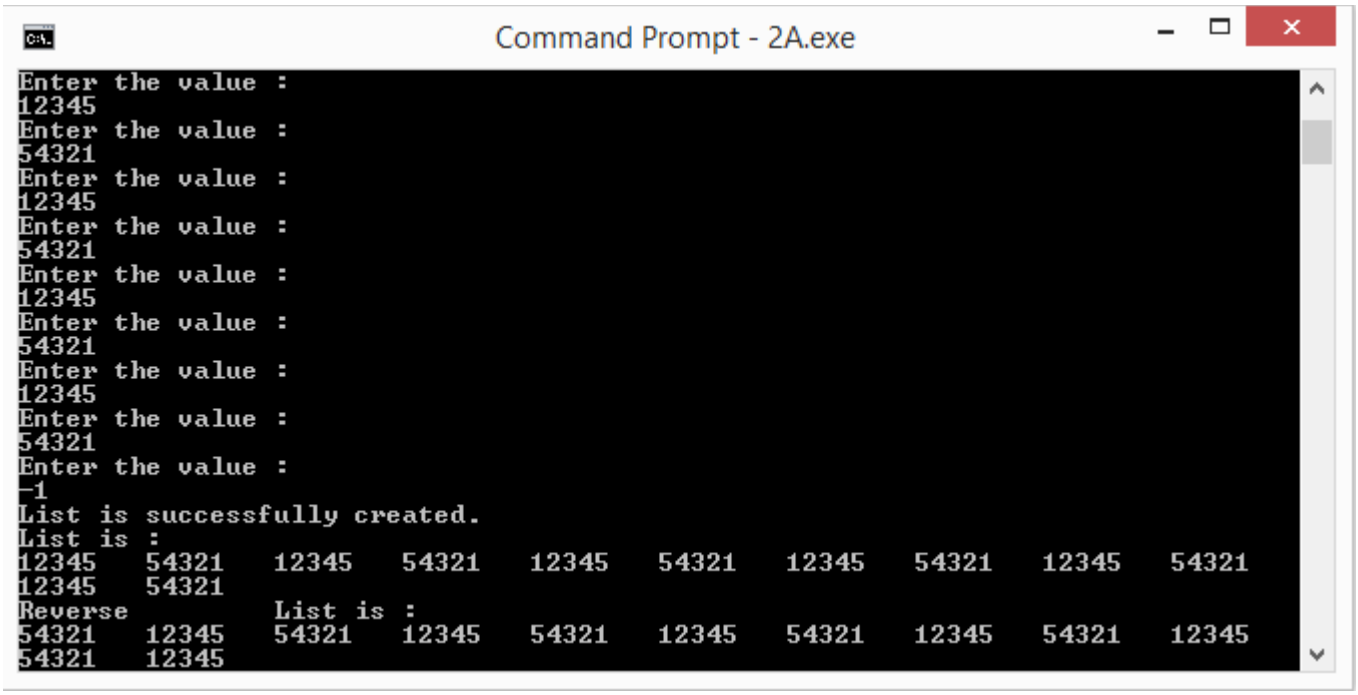
```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
        int info;
        struct node *next;
};
struct node *start=NULL;
struct node *create(struct node *start);
struct node *dispaly(struct node *start);
void reverse(struct node *start);
int main()
{
clrscr();
start=create(start);
start=dispaly(start);
printf("\n");
printf("Reverse \t");
reverse(start);
        getch ();
        return 0;
}
struct node *create(struct node *start)
```

```
{
        struct node *new_node=NULL,*temp=NULL;
        int val;
        printf("Enter -1 value to exit list.\n");
        printf("Enter the value : \n");
        scanf("%d",&val);
        while(val!=-1)
        {
        new_node=(struct node*)malloc(sizeof(struct node));
        new_node->info=val;
        if(start==NULL)
        {
                start=new_node;
                new_node->next=NULL;
        }
        else
        {
                temp=start;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=new_node;
                new_node->next=NULL;
        }
        printf("Enter the value : \n");
        scanf("%d",&val);
}
printf("List is successfully created.\n");
return start;
}
struct node *dispaly(struct node *start)
```

```
{
        struct node *temp=NULL;
        temp=start;
        printf("List is :\n");
        while(temp!=NULL)
        {
                printf("%d \t",temp->info);
                temp=temp->next;
        }
        return start;
}
void reverse(struct node *start)
{
struct node *prev=NULL;
struct node *current=start;
struct node *next_node;
while(current!=NULL)
{
 next_node=current->next;
current->next=prev;
prev=current;
current=next_node;
}
start=prev;
start=dispaly(start);
}
```

OUTPUT:





PRACTICAL 2B

Write a program to search the elements in the linked list and display the same.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
        int info;
        struct node *next;
};
struct node *start=NULL;
struct node *create(struct node *start);
struct node *dispaly(struct node *start);
struct node *search(struct node *start);
int main()
{
start=create(start);
start=dispaly(start);
printf("\n");
start=search(start);
        getch ();
        return 0;
}
struct node *create(struct node *start)
{
        struct node *new_node=NULL,*temp=NULL;
        int val;
        printf("Enter -1 value to exit list.\n");
        printf("Enter the value : \n");
```

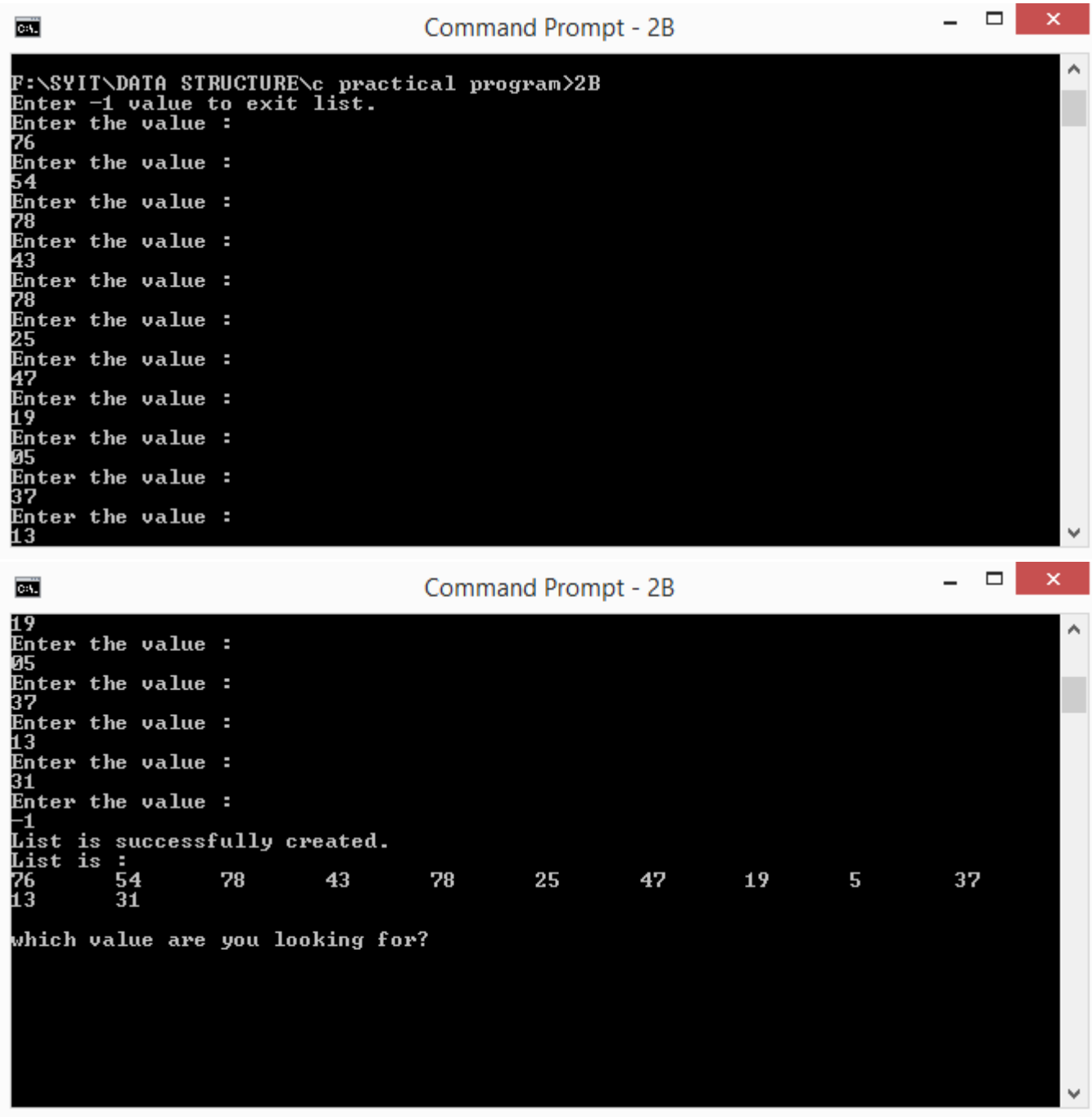
```
        scanf("%d",&val);
        while(val!=-1)
        {
        new_node=(struct node*)malloc(sizeof(struct node));
        new_node->info=val;
        if(start==NULL)
        {
                start=new_node;
                new_node->next=NULL;
        }
        else
        {
                temp=start;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=new_node;
                new_node->next=NULL;
        }
        printf("Enter the value : \n");
        scanf("%d",&val);
}
printf("List is successfully created.\n");
return start;
}
struct node *dispaly(struct node *start)
{
        struct node *temp=NULL;
        temp=start;
        printf("List is :\n");
        while(temp!=NULL)
```

```
        {
                printf("%d \t",temp->info);
                temp=temp->next;
        }
        return start;
}
struct node *search(struct node *start)
{
int  val,count;
struct node *temp;
printf("\nwhich value are you looking for?\n");
scanf("%d",&val);
count=1;
temp=start;
while(temp->info!=val && temp->next!=NULL)
{
temp=temp->next;
count++;
}
//temp=temp->next;
if(temp->next==NULL && temp->info!=val)
{
printf("value not found");
}
else if(temp->next==NULL && temp->info==val)
{
printf("value found at %d node",count);
}
else
{
printf("value found at %d node",count);
}
```

return start;

}

OUTPUT:



PRACTICAL 2C

Write a program to create double linked list and sort the elements in the linked list.

INPUT:

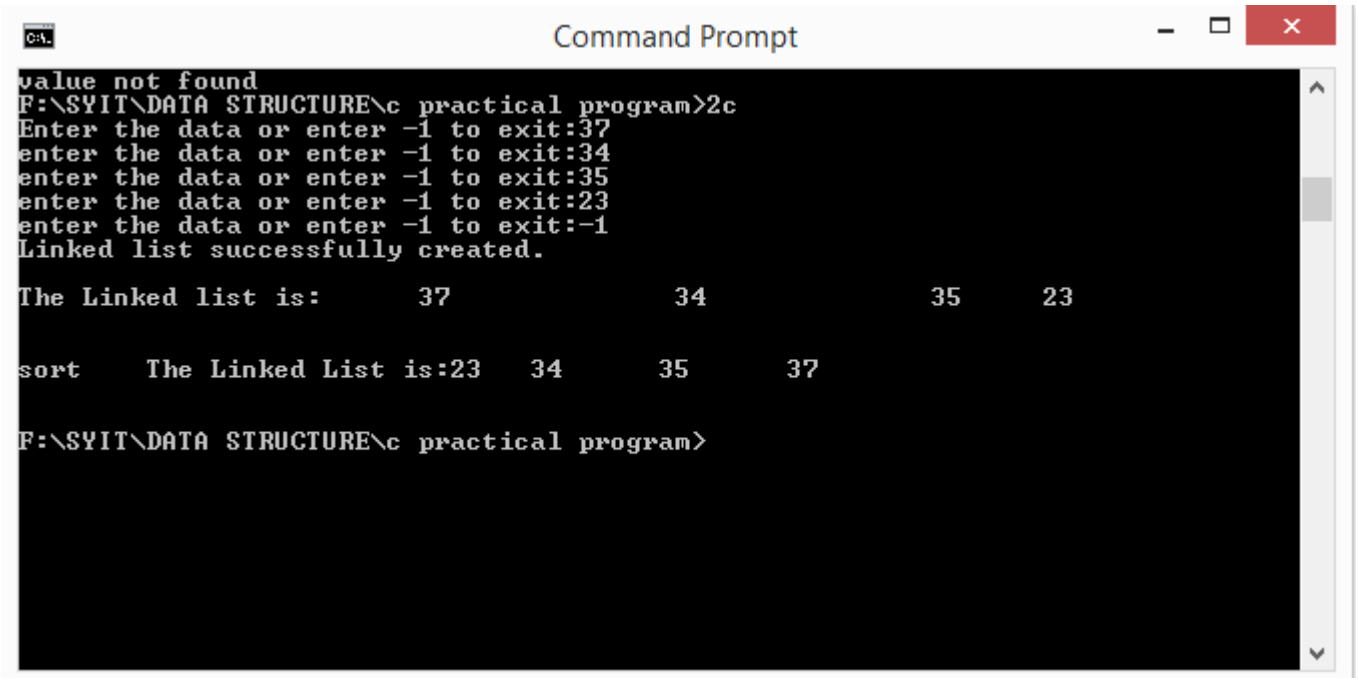
```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{int data;
        struct node *next;
        struct node *prev;
};
struct node *start=NULL;
struct node *create(struct node *start);
struct node *display(struct node *start);
struct node *sort(struct node *start);
int main()
{
        start=create(start);
        start=display(start);
        printf("\n");
        printf("sort \t");
        start=sort(start);
}
struct node *create(struct node *start)
{
        struct node *new_node=NULL,*temp=NULL,prev;
        int val;
        printf("Enter the data or enter -1 to exit:");
        scanf("%d",&val);
```

```
	while(val!=-1)
	{
		new_node=(struct node*)malloc(sizeof(struct node));
		new_node->data=val;
		if(start==NULL)
		{
			start=new_node;
			new_node->next=NULL;
			new_node->prev=NULL;
		}
		else
		{
			temp=start;
			while(temp->next!=NULL)
			{
				temp=temp->next;
			}
			temp->next=new_node;
			new_node->prev=NULL;
			new_node->next=NULL;
		}
		printf("enter the data or enter -1 to exit:");
		scanf("%d",&val);
	}
	printf("Linked list successfully created.\n");
	return start;
}
struct node *display(struct node *start)
{
	struct node *temp=NULL;
	temp=start;
	printf("\nThe Linked list is:");
```

```
        while(temp->next!=NULL)
        {
                printf("\t %d \t",temp->data);
                temp=temp->next;
        }
        if(temp->next==NULL)
        printf("%d \n",temp->data);
        printf("\n");
        return start;
}
struct node *sort(struct node*start)
{
struct node *temp1=start;
struct node *temp2,*temp;
int x;
while (temp1->next!=NULL)
{
        temp2=start;
        while(temp2->next!=NULL)
        {
                temp=temp2->next;
                if(temp2->data>temp->data)
                {
                        x=temp->data;
                        temp->data=temp2->data;
                        temp2->data=x;
                }
                temp2=temp2->next;
        }
        temp1=temp1->next;
}
temp=start;
```

```
printf("The Linked List is:");
while(temp->next!=NULL)
{
        printf("%d \t",temp->data);
        temp=temp->next;
}
if(temp->next==NULL)
printf("%d \n",temp->data);
printf("\n");
return start;
}
```

OUTPUT:



Implement the following for Stack:

PRACTICAL 3A

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

INPUT:

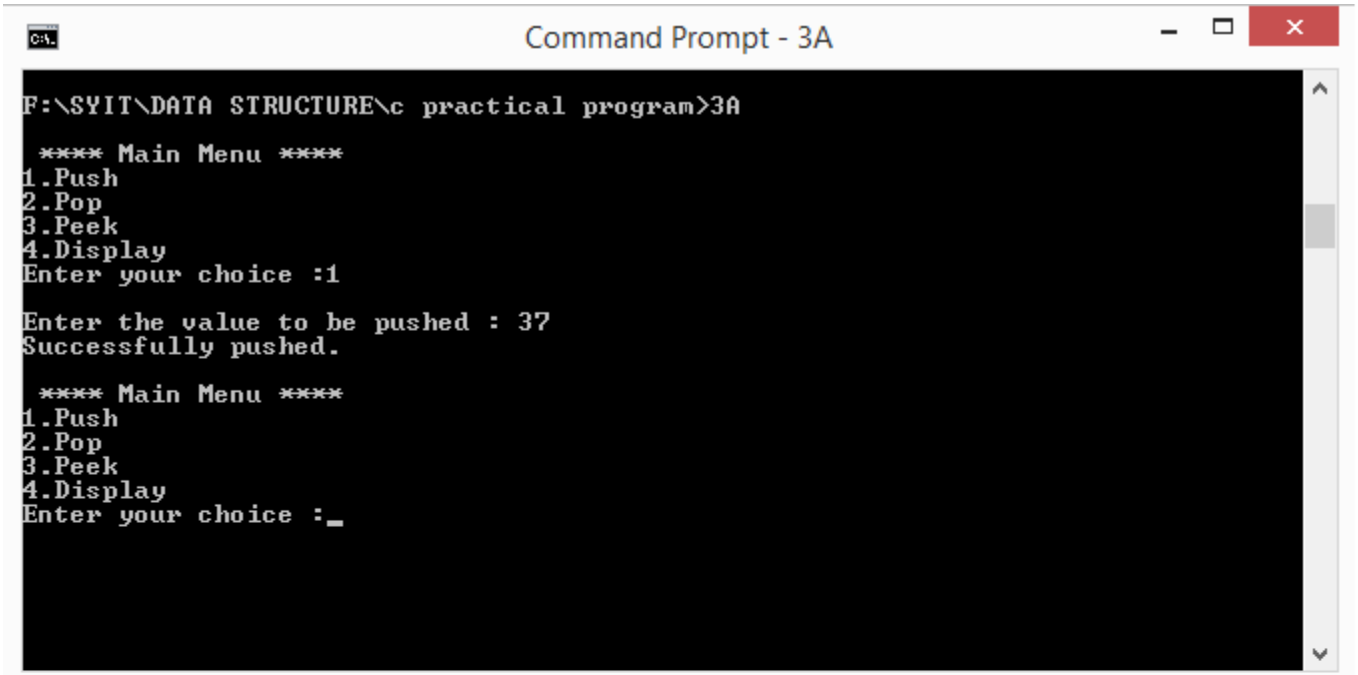
```
#include<stdio.h>
#include<conio.h>
#define MAX 30
int stack[MAX];
int top =-1; //Stack is empty.
void push();
int pop();
int peek();
void display();
int main()
{
int choice;
do
{
printf("\n **** Main Menu **** \n");
printf("1.Push\n");
printf("2.Pop \n");
printf("3.Peek \n");
printf("4.Display \n");
printf("Enter your choice :");
scanf("%d",&choice);
printf("\n");
switch(choice)
{
case 1: push();
break;
```

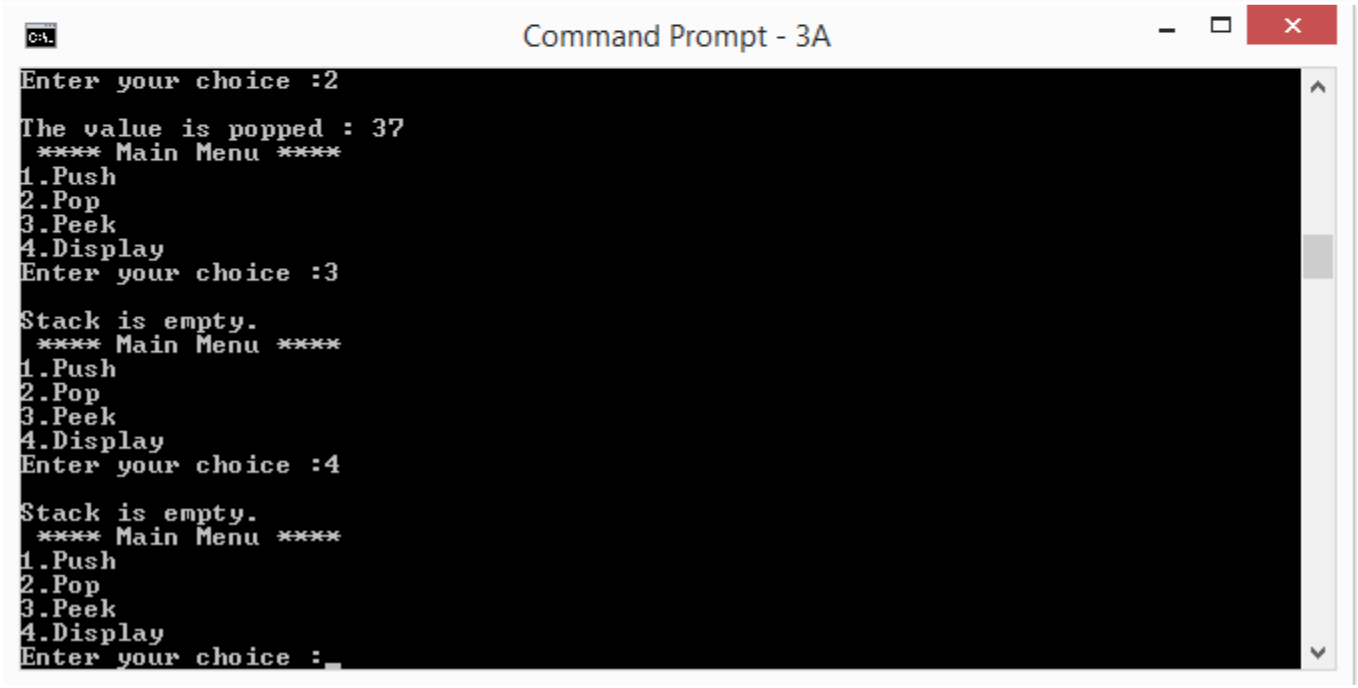
```
case 2 : pop();
break;
case 3 : peek();
break;
case 4 : display();
break;
case 5 : break;
}
}
while(choice!=5);
return 0;
}
void push()
{
int val;
if(top == MAX -1)
{
printf("Stack is full.");
}                    else
{
printf("Enter the value to be pushed : ");
scanf("%d",&val);
stack[++top]=val;
printf("Successfully pushed.\n");
}
}
int pop()
{
if(top == -1)
{
printf("Stack is already empty.");
}
```

```
else
{
int val = stack[top];
top--;
printf("The value is popped : %d",val);
}
}
int peek()
{
if(top == -1)
{
printf("Stack is empty.");
}
else
{
int topmost = stack[top];
printf("The topmost element of stack : %d ",topmost);
}
}
void display()
{
if(top == -1)
{
printf("Stack is empty.");
}
else
{
int i;
printf("Stack is : ");
for(i=top;i>=0;i--)
{
printf("\t%d",stack[i]);
```

```
}
}
}
```

OUTPUT:





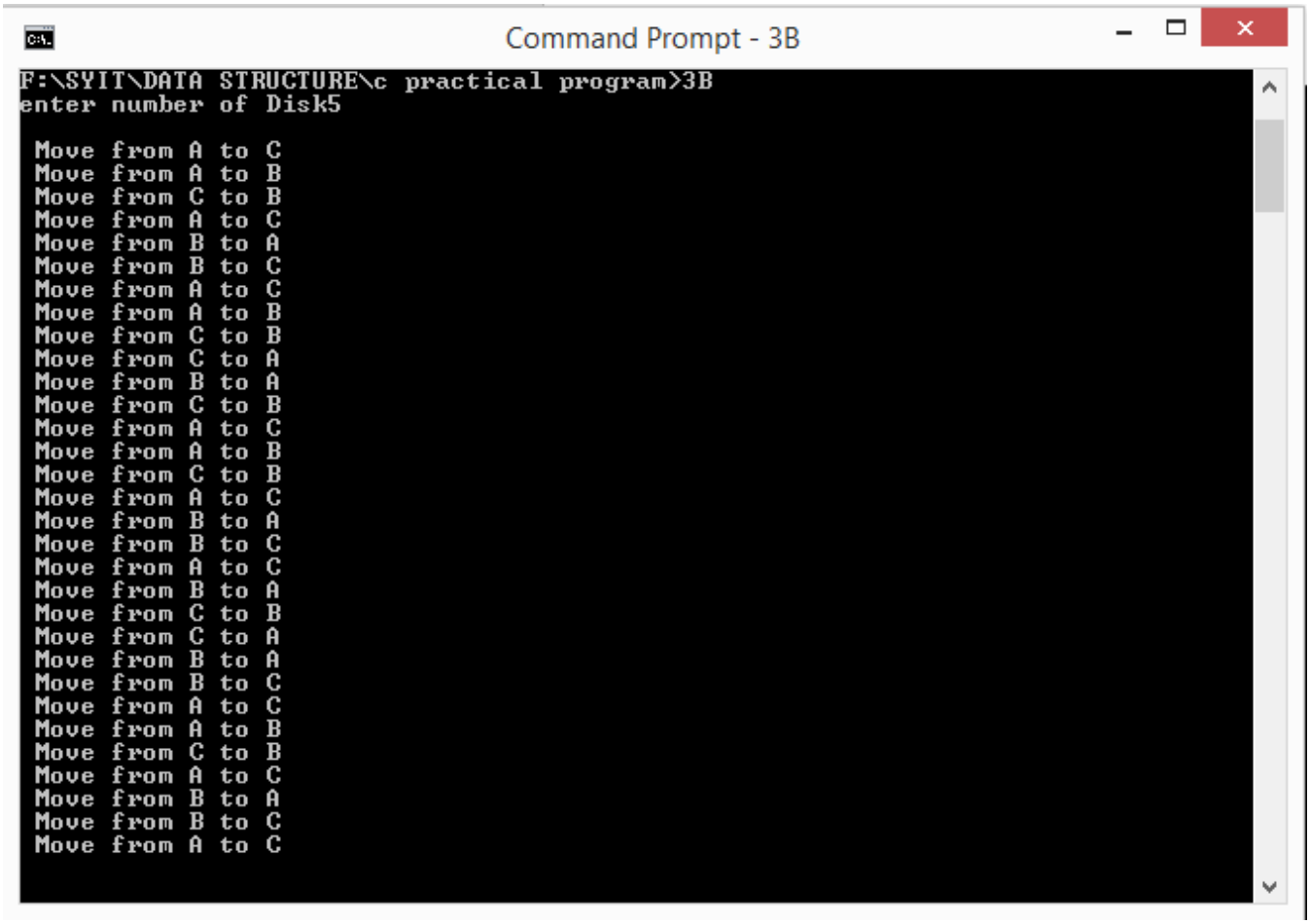
PRACTICAL 3B

Write a program to convert an infix expression to postfix and prefix conversion.

INPUT:

```
#include<stdio.h>
#include<conio.h>
int move(int n,char source,char temp,char destination);
int main()
{
int n;
printf("enter number of Disk");
scanf("%d",&n);
move(n,'A','B','C');
getch();
return 0;
}
int move(int n,char source,char temp,char destination)
{
if(n == 1)
{
printf("\n Move from %c to %c",source,destination);
}
else
{
move(n-1,source,destination,temp);
move(1,source,temp,destination);
move(n-1,temp,source,destination);
}
}
```

Output:



PRACTICAL 3C

Write a program to implement Tower of Hanoi problem.

INPUT:

```
#include<stdio.h>
#include<conio.h>
int move(int n,char source,char temp,char destination);
int main()
{
int n;
printf("enter number of Disk");
scanf("%d",&n);
move(n,'A','B','C');
getch();
return 0;
}
int move(int n,char source,char temp,char destination)
{
if(n == 1)
{
printf("\n Move from %c to %c",source,destination);
}
else
{
move(n-1,source,destination,temp);
move(1,source,temp,destination);
move(n-1,temp,source,destination);
}
}
```

OUTPUT:

Implement the following for Queue:

PRACTICAL 4A

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

INPUT:

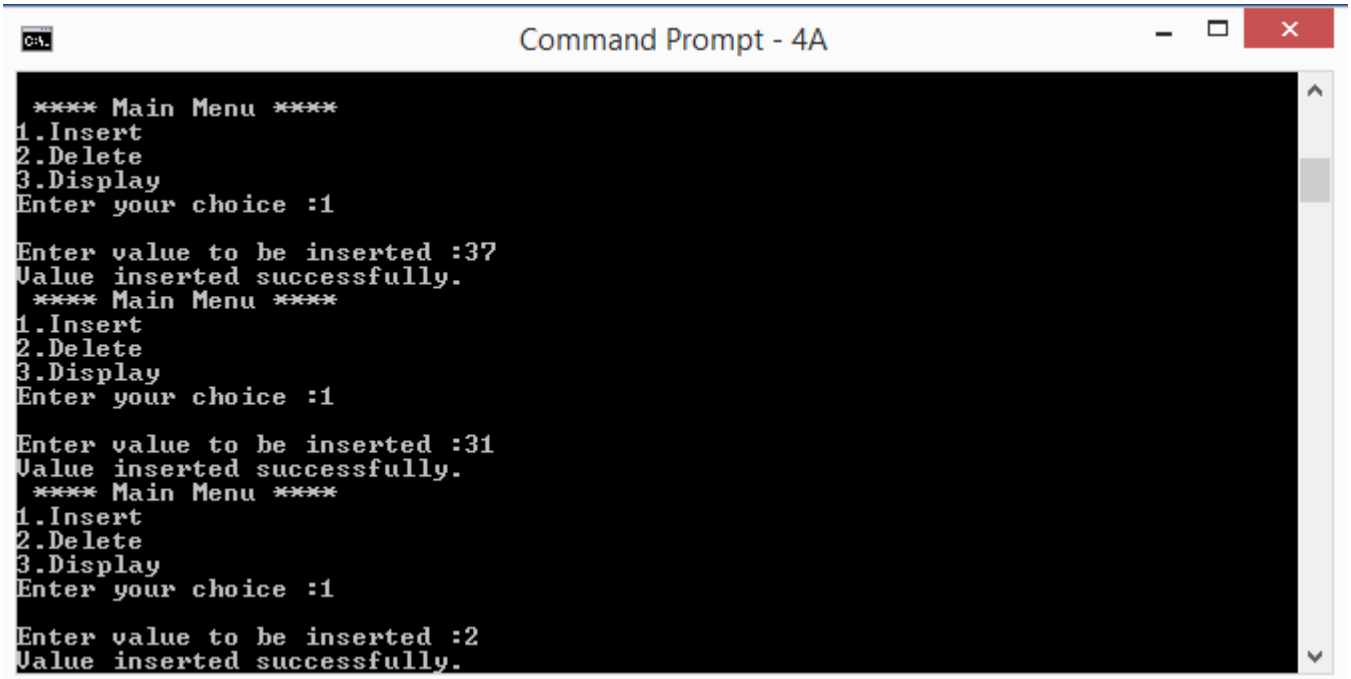
```
#include<stdio.h>
#include<conio.h>
#define max  30
int rear=-1;
int front=-1;
void insert();
int deleteq();
void display();
int q[max];
void insert()
{
int val;
printf("Enter value to be inserted :");
scanf("%d",&val);
if(rear==max-1)
{
printf("Queue is full.");
}
else if(front==-1)
{
front=rear=0;
q[rear]=val;
printf("Value inserted successfully.");
}
else
{
```

```
q[++rear]=val;
printf("Value inserted successfully.");
}
}
int deleteq()
{
if(front==-1)
{
printf("Queue is already empty.");
return -1;
}
else if(front==rear) //Only one item is present.
{          int val ;
val=q[front];
front=rear=-1;
printf("Value to be deletede : %d",val);
return val;
}
else
{
int val ;
val=q[front];
front++;
printf("Value to be deletede : %d",val);
return val;
}
}
void display()
{
if(front==-1)
{
printf("Queue is empty.");
```

```
}
else
{
int i;
printf("Queue is :");
for(i=front;i<=rear;i++)
{
printf("%d",q[i]);
}
}
}
int main()
{
int choice;
do
{
printf("\n **** Main Menu **** \n");
printf("1.Insert\n");
printf("2.Delete\n");
printf("3.Display \n");
printf("Enter your choice :");
scanf("%d",&choice);
printf("\n");
switch(choice)
{
case 1: insert();
break;
case 2 : deleteq();
break;
case 3 : display();
break;
case 4 : break;
```

```
}
}
while(choice!=4);
return 0;
}
```

OUTPUT:



PRACTICAL 4B

Write a program to implement the concept of Circular Queue.

INPUT:

```
/*
 * C++ Program to Implement Circular Queue
 */
#include <iostream>
#define MAX 5
using namespace std;
/*
 * Class Circular Queue
 */
class Circular_Queue
{
   private:
      int *cqueue_arr;
      int front, rear;
   public:
      Circular_Queue()
      {
         cqueue_arr = new int [MAX];
         rear = front = -1;
      }
      /*
       * Insert into Circular Queue
       */
      void insert(int item)
      {
         if ((front == 0 && rear == MAX-1) || (front == rear+1))
         {
            cout<<"Queue Overflow \n";
```

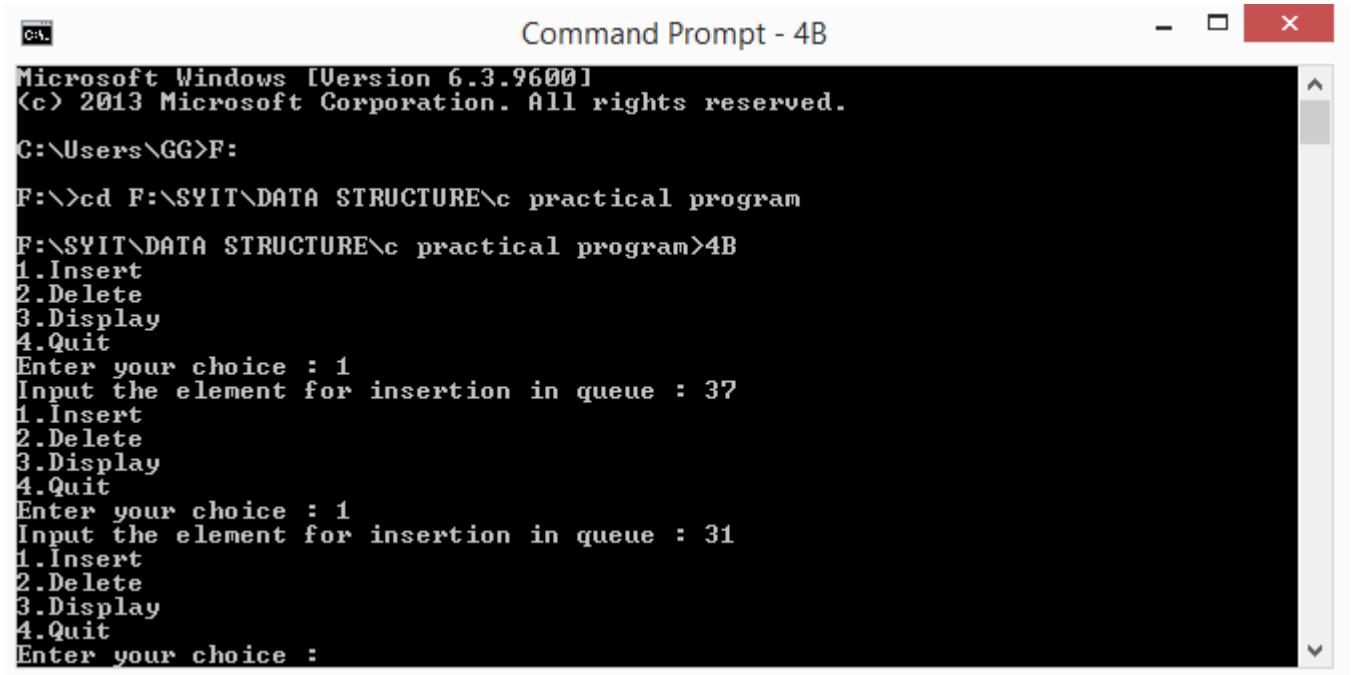
```
        return;
    }
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }
    else
    {
        if (rear == MAX - 1)
            rear = 0;
        else
            rear = rear + 1;
    }
    cqueue_arr[rear] = item ;
}
/*
 * Delete from Circular Queue
 */
void del()
{
    if (front == -1)
    {
        cout<<"Queue Underflow\n";
        return ;
    }
    cout<<"Element deleted from queue is : "<<cqueue_arr[front]<<endl;
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
```

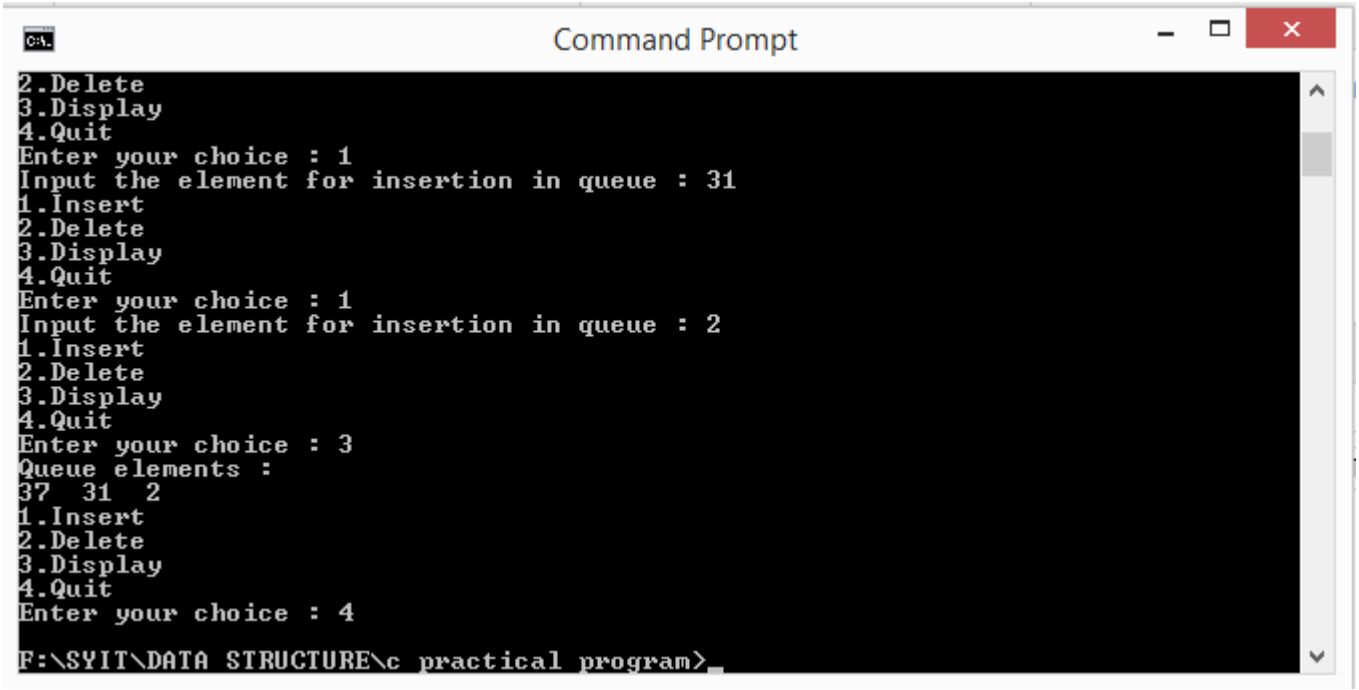
```
    else
    {
        if (front == MAX - 1)
            front = 0;
        else
            front = front + 1;
    }
}
/*
 * Display Circular Queue
 */
void display()
{
    int front_pos = front, rear_pos = rear;
    if (front == -1)
    {
        cout<<"Queue is empty\n";
        return;
    }
    cout<<"Queue elements :\n";
    if (front_pos <= rear_pos)
    {
        while (front_pos <= rear_pos)
        {
            cout<<cqueue_arr[front_pos]<<" ";
            front_pos++;
        }
    }
    else
    {
        while (front_pos <= MAX - 1)
        {
```

```
            cout<<cqueue_arr[front_pos]<<" ";
            front_pos++;
          }
          front_pos = 0;
          while (front_pos <= rear_pos)
          {
            cout<<cqueue_arr[front_pos]<<" ";
            front_pos++;
          }
        }
        cout<<endl;
      }
};
/*
 * Main
 */
int main()
{
   int choice, item;
   Circular_Queue cq;
   do
   {
      cout<<"1.Insert\n";
      cout<<"2.Delete\n";
      cout<<"3.Display\n";
      cout<<"4.Quit\n";
      cout<<"Enter your choice : ";
      cin>>choice;
      switch(choice)
      {
      case 1:
         cout<<"Input the element for insertion in queue : ";
```

```
            cin>>item;
            cq.insert(item);
                break;
             case 2:
            cq.del();
                break;
          case 3:
            cq.display();
                break;
             case 4:
                break;
             default:
                cout<<"Wrong choice\n";
             }/*End of switch*/
      }
      while(choice != 4);
      return 0;
}
```

OUTPUT:





PRACTICAL 4C

Write a program to implement the concept of Deque.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#define max 5
int front = -1;
int rear = -1;
int insert_rear();
int insert_front();
void display();
int deleteq_rear();
int deleteq_front();
int q[max];
int main()
{       int choice;
        clrscr();
        do
        {
                printf("\n **** Main Menu **** \n");
                printf("1.Insert From Rear\n");
                printf("2.Insert From Front\n");
                printf("3.Delete From Front \n");
                printf("4.Delete From Rear \n");
                printf("5.Display\n");
                printf("Enter your choice :");
                scanf("%d",&choice);
                printf("\n");
                switch(choice)
                {
                        case 1: insert_rear();
```

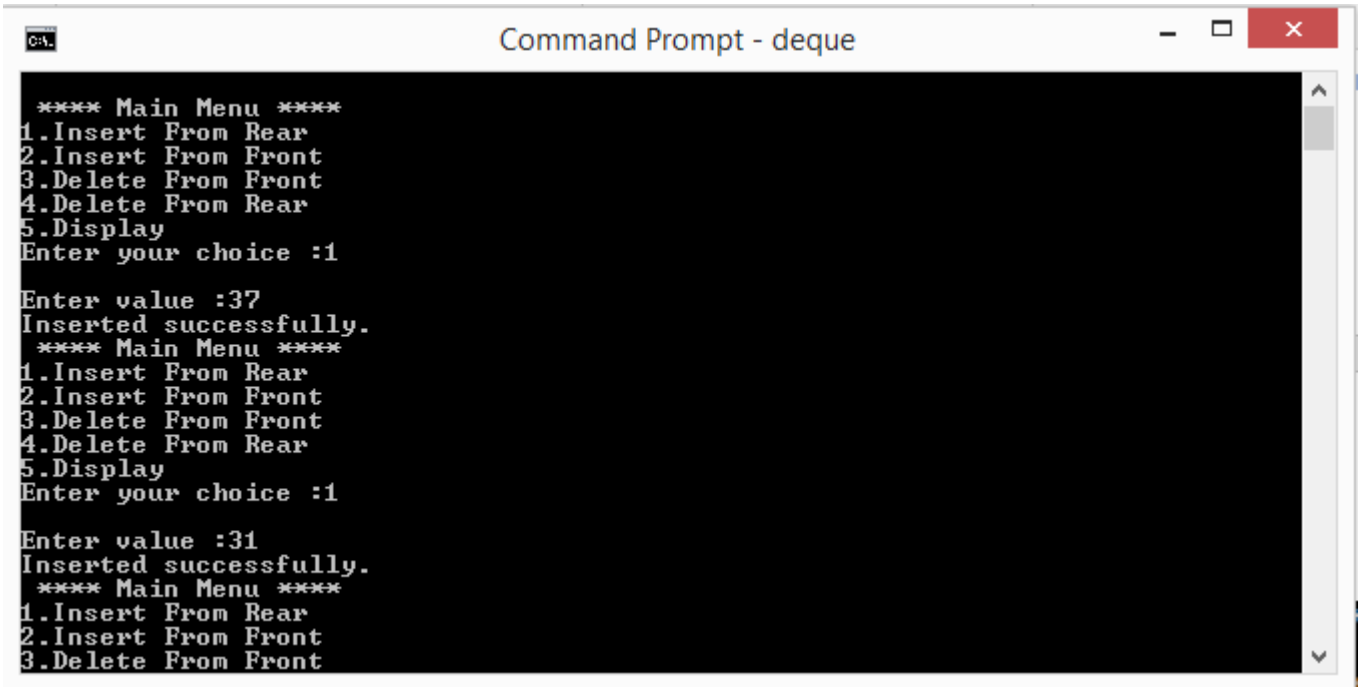
```
                        break;
                        case 2 : insert_front();
                        break;
                        case 3 : deleteq_front();
                        break;
                        case 4 : deleteq_rear();
                        break;
                        case 5 : display();
                        break;
                        case 6 : break;
                        }
        }
        while(choice!=6);
        return 0;
}
int insert_rear()
{       int val;
        printf("Enter value :");
        scanf("%d",&val);
        if((rear+1)%max==front)
        {
                printf("Queue is full.");
                return 0;
        }
        else if(rear==-1)
        {
                rear=front=0;
                q[rear]=val;
                printf("Inserted successfully.");
                return val;
        }
        else
```

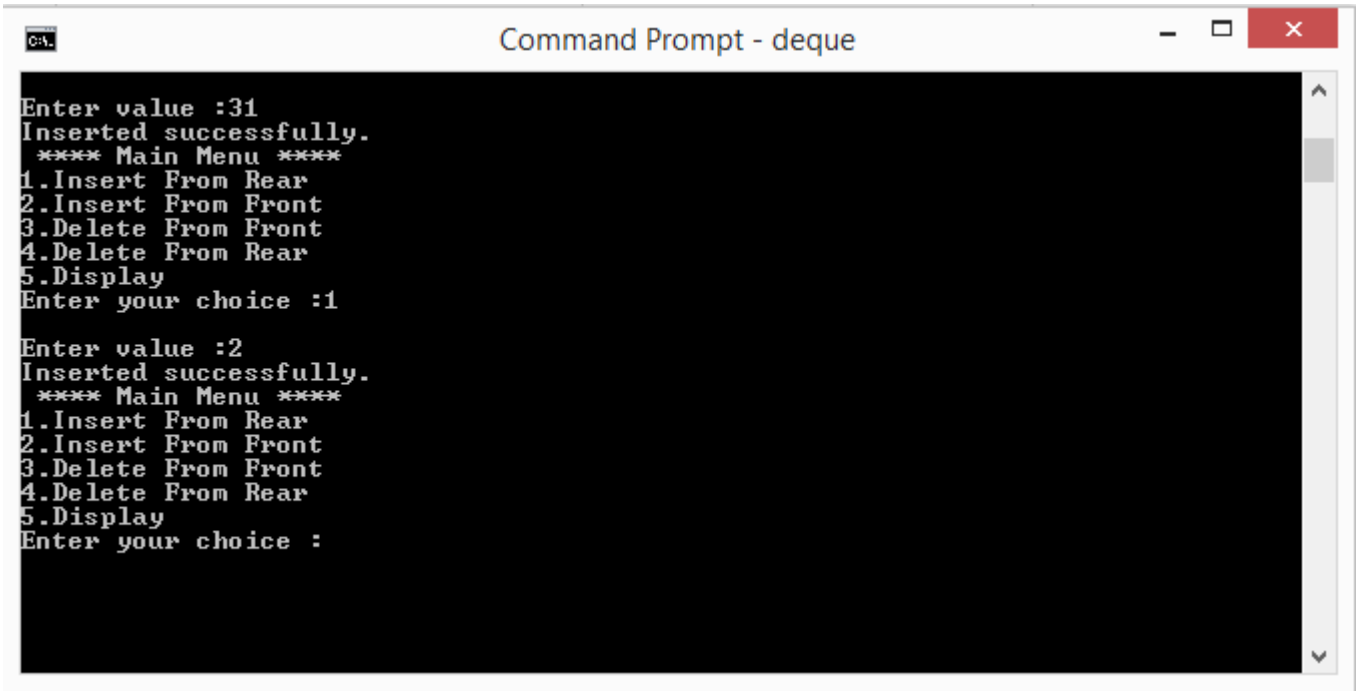
```
        {
                rear=(rear + 1)%max;
                q[rear]=val;
                printf("Inserted successfully.");
                return val;
        }
}
int insert_front()
{       int val;
        printf("Enter value :");
        scanf("%d",&val);
        if((rear+1)%max==front)
        {
                printf("Queue is full.");
                return 0;
        }
        else if(front==-1)
        {
                rear=front=0;
                q[front]=val;
                printf("Inserted successfully.");
                return val;
        }
        else
        {
           front=(front-1+max)%max;
                q[front]=val;
                printf("Inserted successfully.");
                return val;
        }
}
int deleteq_front()
```

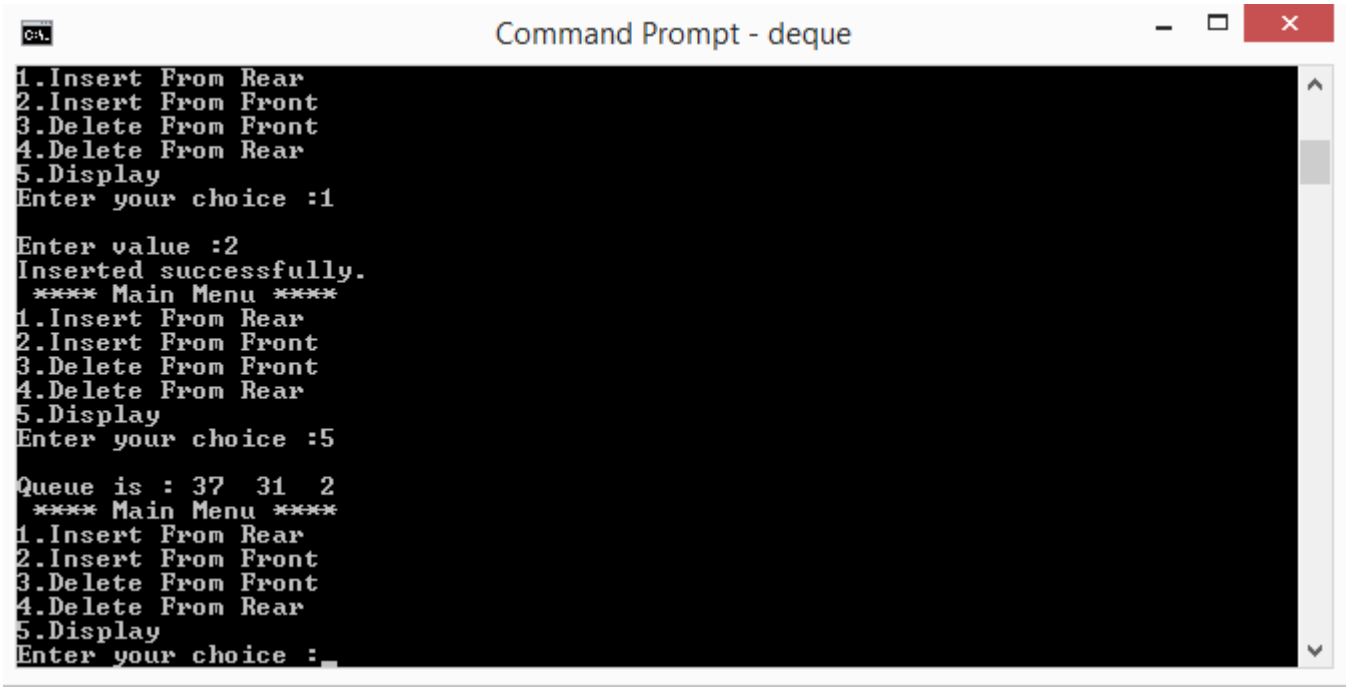
```
{       int val;
        if(front== -1)
        {
                printf("Queue is empty.");
            //  return -1;
        }
        else if(front == rear)
        {
                int val=q[front];
                front=rear= -1;
                printf("Deleted value : %d",val);
                return val;
        }
        else
        {
           val=q[front];
                front=(front+1)%max;
                printf("Deleted value : %d",val);
                return val;
        }
}
int deleteq_rear()
{       int val;
        if(rear== -1)
        {
                printf("Queue is empty.");
                return -1;
        }
        else if(front == rear)
        {
                int val=q[rear];
                front=rear= -1;
```

```
                printf("Deleted value : %d",val);
                return val;
        }
        else
        {
                val=q[rear];
           rear=(rear-1+max)%max;
                printf("Deleted value : %d",val);
                return val;
        }
}
void display()
{       int i;
        if(front ==-1)
        {
                        printf("Queue is empty.");

        }
        else
        {

                printf("Queue is :"); for(i=front;i!
                =rear;i=(i+1)%max)
                {
                        printf(" %d ",q[i]);
                }
                printf(" %d ",q[i]);
        }

}
```

OUTPUT:







Implement the following sorting techniques:

PRACTICAL 5A

Write a program to implement bubble sort.

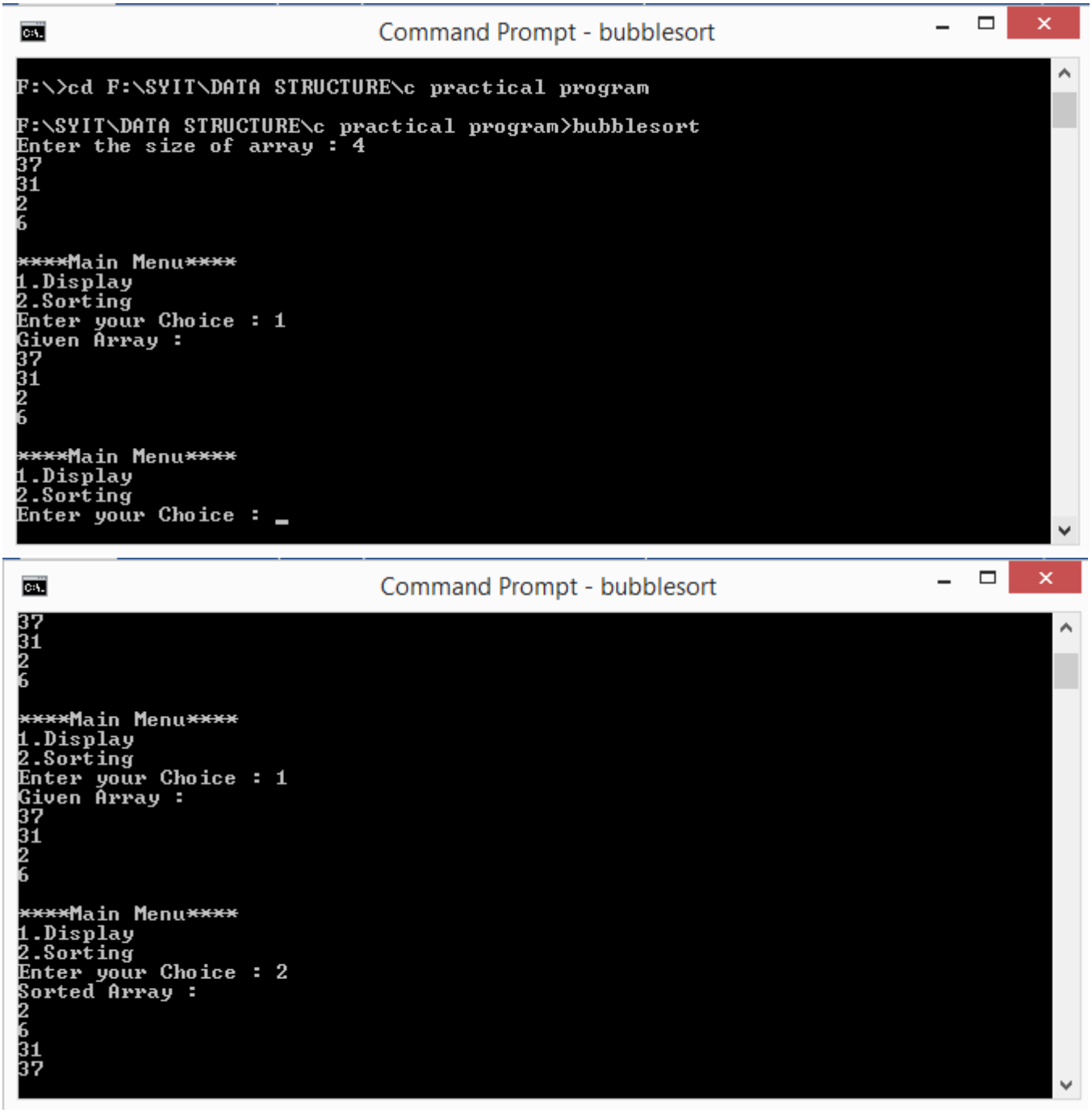
INPUT:

```
#include<stdio.h>
#include<conio.h>
int size,val;
void disp(int size);
int sort(int size);
int arr[20];
int main()
{
  int i,ch;
  printf("Enter the size of array : ");
  scanf("%d",&size);
  for(i=0;i<size;i++)
  {
        scanf("%d",&arr[i]);
  }
  do
  {
        printf("\n****Main Menu****\n");
        printf("1.Display\n");
        printf("2.Sorting\n");
       printf("Enter your Choice : ");
       scanf("%d",&ch);
        switch(ch)
        {
              case 1:disp(size);
              break;
          case 2:sort(size);
```

```
                break;
            }
    }
    while(ch!=2);
getch ();
return 0;
}
void disp(int size)
{
        int i;
        printf("Given Array :\n");
        for(i=0;i<size;i++)
        {
                printf("%d\n",arr[i]);
        }
}
int sort(size)
{
        int i,j;
        for(i=0;i<size;i++)
        {
                for(j=0;j<size-i-1;j++)
                {
                        if(arr[j]>arr[j+1])
                        {
                                int temp;
                                temp=arr[j];
                                arr[j]=arr[j+1];
                                arr[j+1]=temp;
                        }
                }
        }
```

```
        printf("Sorted Array : \n");
        for(i=0;i<size;i++)
        {
                printf("%d \n",arr[i]);
        }
}
```

OUTPUT:



PRACTICAL 5B

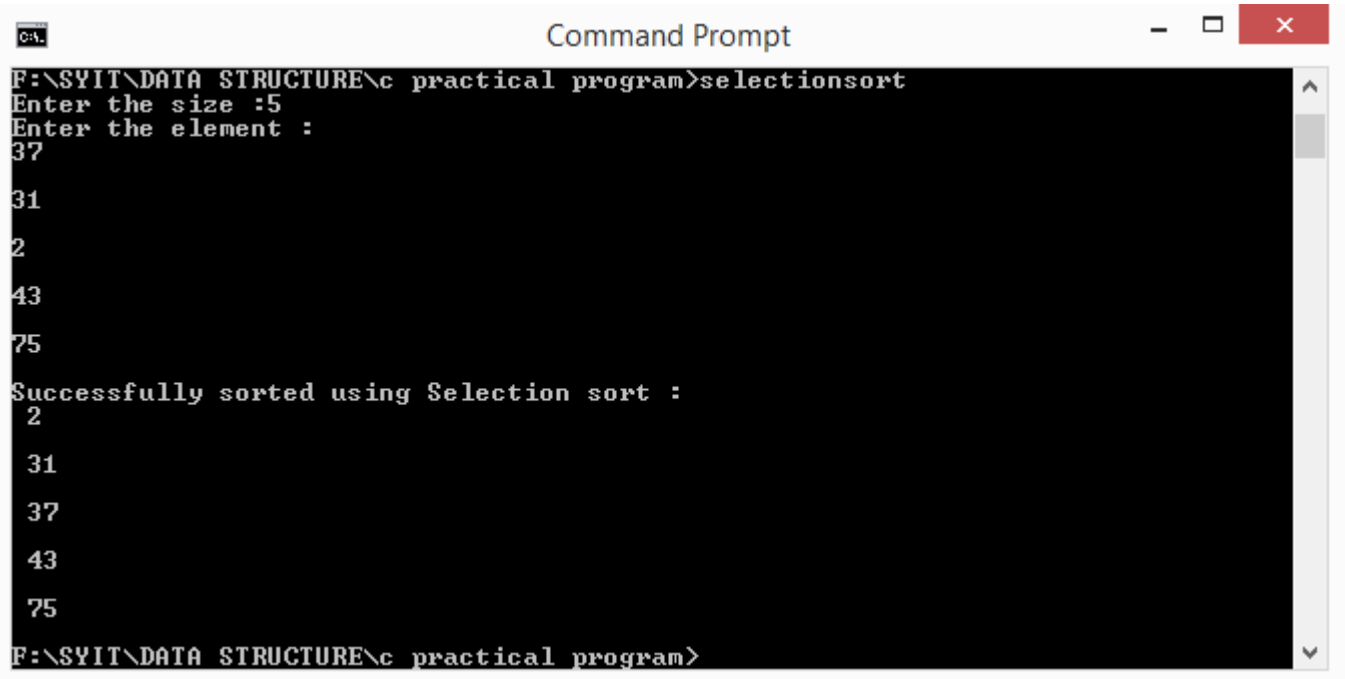
Write a program to implement selection sort.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
int selection_sort(int n);
int A[20];
int selection_sort(int n)
{
        int imin,i,j,temp;
        for(i=0;i<n;i++)
        {
                imin=i;
                for(j=i+1;j<n;j++)
                {
                        if(A[imin]>A[j])
                        {
                                imin=j;
                        }
          }
          temp  = A[i];
          A[i] = A[imin];
          A[imin] = temp;

        }

         printf("Successfully sorted using Selection sort :");
}
int main()
{
        int n,i;
        printf("Enter the size :");
```

```
        scanf("%d",&n);
        printf("Enter the element :\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&A[i]);
                printf("\n");
        }

        selection_sort(n);
                for(i=0;i<n;i++)
        {
                printf("\n %d\n",A[i]);
        }
        return 0;
}
```

OUTPUT:



PRACTICAL 5C

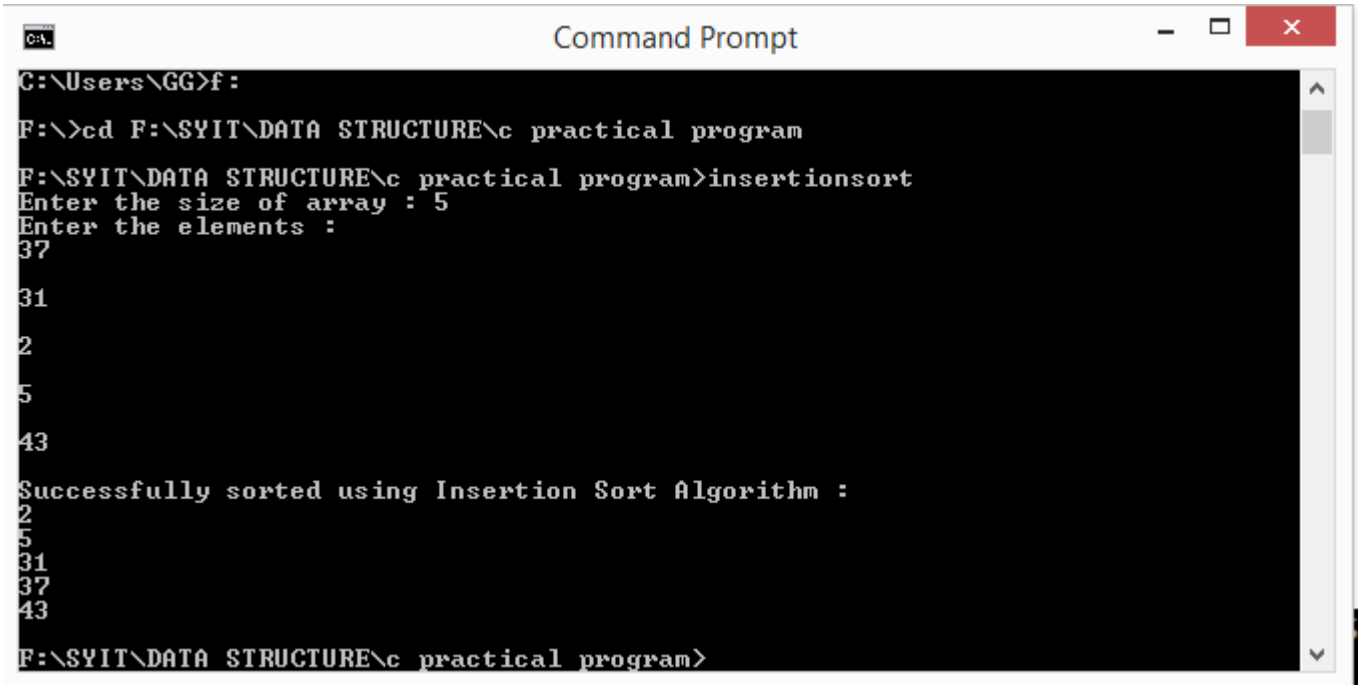
Write a program to implement insertion sort.

INPUT:

```
#include<stdio.h>
#include<conio.h>
int A[10];
void insertion_sort(int n)
{
        int val,vacant,i;
        for(i=1;i<n;i++)
        {
                val=A[i];
                vacant=i;
                while(A[vacant-1]>val && vacant!=0)
                {
                        A[vacant]=A[vacant-1];
                        vacant=vacant - 1;
        }
    A[vacant]=val;
        }
        printf("Successfully sorted using Insertion Sort Algorithm : \n");
}
int main()
{
        int n,i;
        printf("Enter the size of array : ");
        scanf("%d",&n);
        printf("Enter the elements :\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&A[i]);
```

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

        }
}
```

OUTPUT:



Implement the following data structure techniques:

PRACTICAL 6A

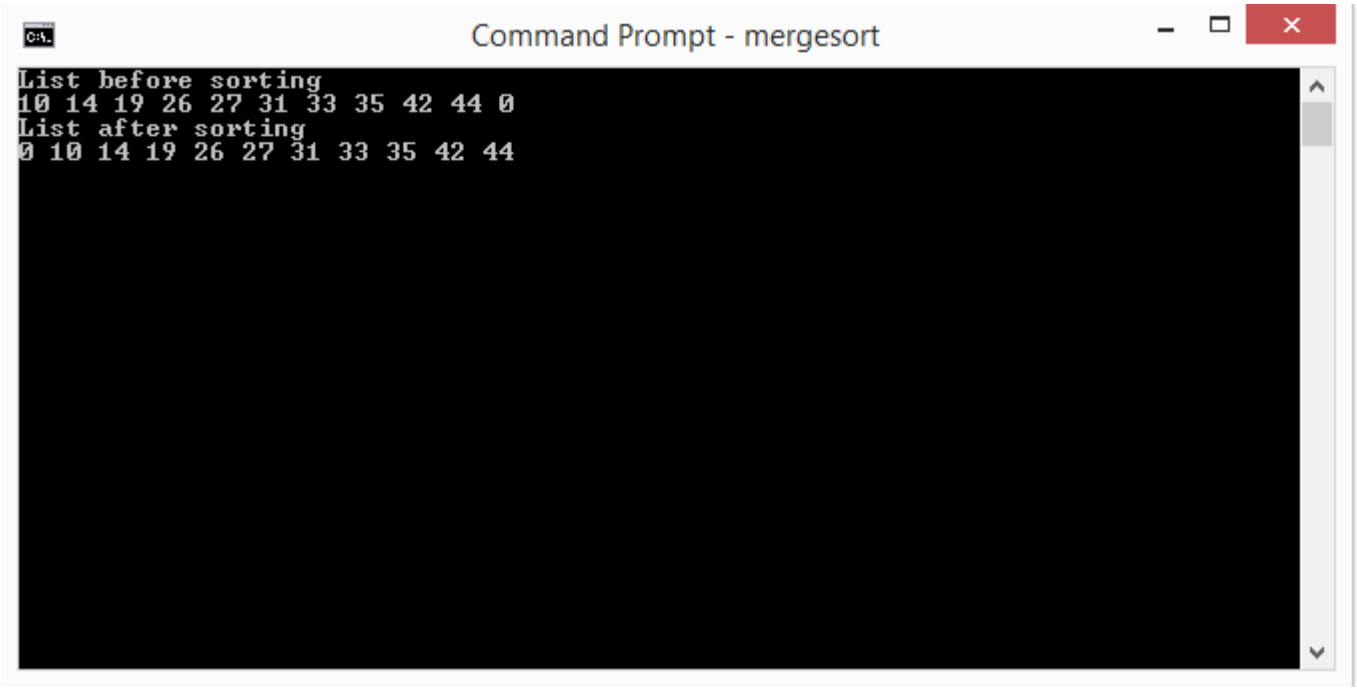
Write a program to implement merge sort.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#define max 10
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;
  }
}
int 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();
}
```

OUTPUT:



PRACTICAL 6B

Write a program to search the element using sequential search.

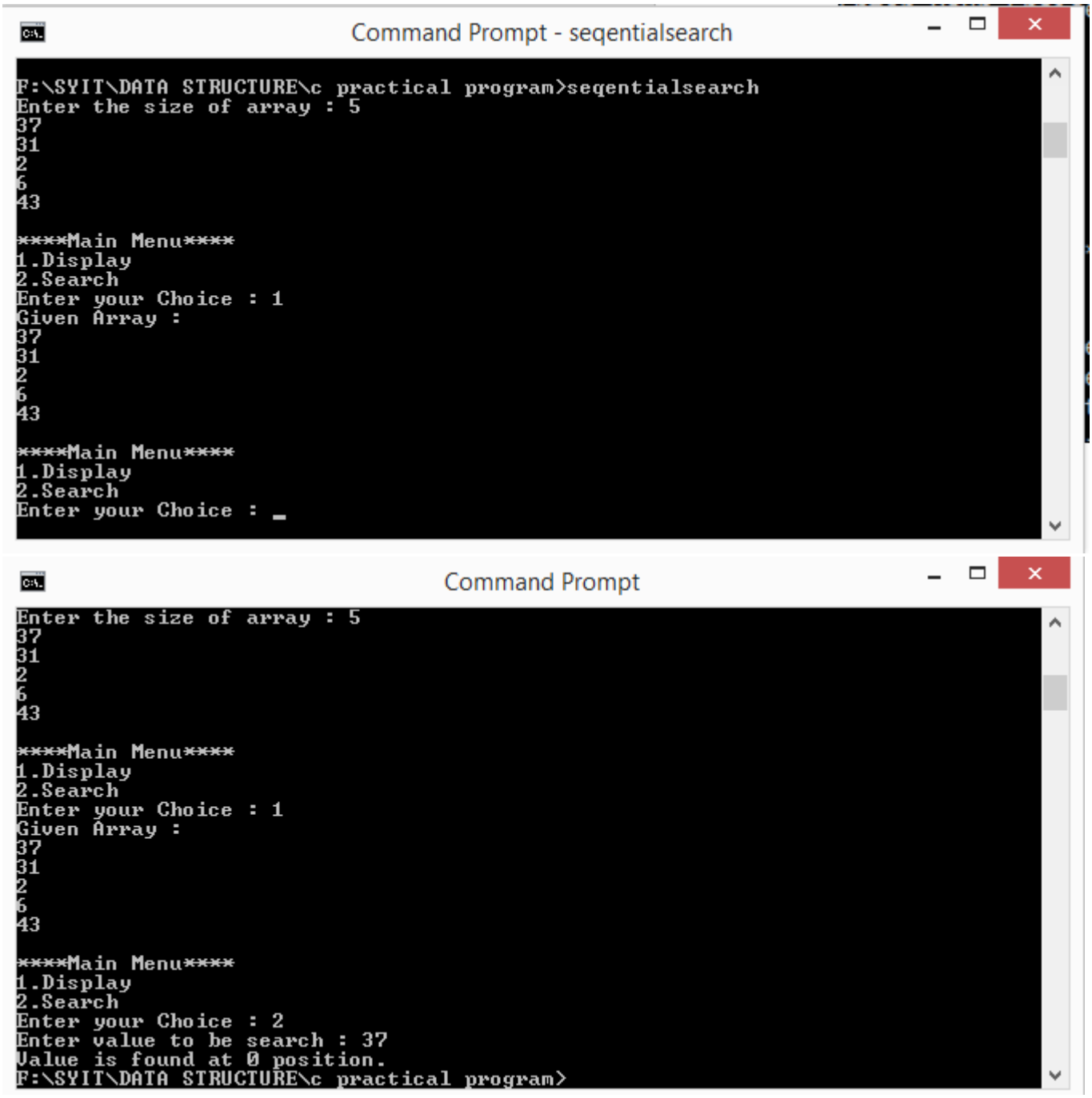
INPUT:

```
#include<stdio.h>
#include<conio.h>
int size,val;
void disp(int size);
void search(int val,int size);
int arr[20];
int main()
{
  int i,ch;
  printf("Enter the size of array : ");
  scanf("%d",&size);
  for(i=0;i<size;i++)
  {
        scanf("%d",&arr[i]);
  }
  do
  {
        printf("\n****Main Menu****\n");
        printf("1.Display\n");
        printf("2.Search\n");
       printf("Enter your Choice : ");
       scanf("%d",&ch);
        switch(ch)
        {
              case 1:disp(size);
              break;
              case 2:printf("Enter value to be search : ");
              scanf("%d",&val);
```

```
                search(val,size);
                  break;
            }
   }
   while(ch!=2);
getch ();
return 0;
}
void search(int val,int size)
{
        int i;
        for(i=0;i<size;i++)
        {
         if(arr[i]==val)
         {
                printf("Value is found at %d position.",i);
                break;
          }
        }
        if(i==size)
        {
                printf("Value is not found.");
        }
}
void disp(int size)
{
        int i;
        printf("Given Array :\n");
        for(i=0;i<size;i++)
        {
                printf("%d\n",arr[i]);
        }
```

}

OUTPUT:



PRACTICAL 6C

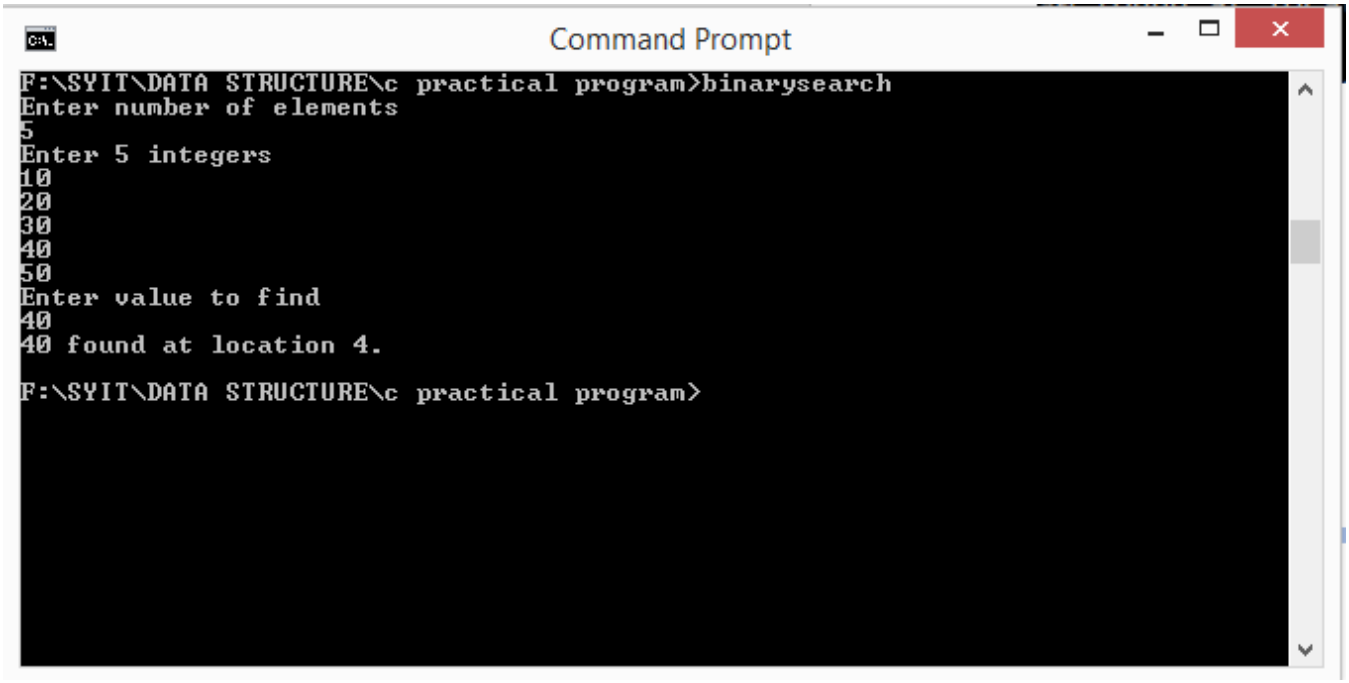
Write a program to search the element using binary search.

INPUT:

```
#include <stdio.h>

int main()
{
  int c, first, last, middle, n, search, array[100];

  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 if(array[middle]>search)
      last = middle - 1;

    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d is not present in the list.\n", search);

  return 0;
}
```

OUTPUT:



Implement the following data structure techniques:

PRACTICAL 7

A .Write a program to create the tree and display the elements.

B . Write a program to construct the binary tree.

C . Write a program for inorder, postorder and preorder traversal of tree

INPUT:

```
#include<stdio.h>
#include<malloc.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *root=NULL;
struct node *create(struct node*);
struct node *display(struct node*);
void preorder(struct node *temp);
void postorder(struct node *temp);
void inorder(struct node *temp);
int main()
{
int choice, val, count, min, max;
do
{
printf("**** Main Menu ***\n");
printf("1. create a binary search\n");
printf("2. Display the tree \n");
printf("3. EXIT \n");
printf("Enter your choice:");
scanf("%d",&choice);
```

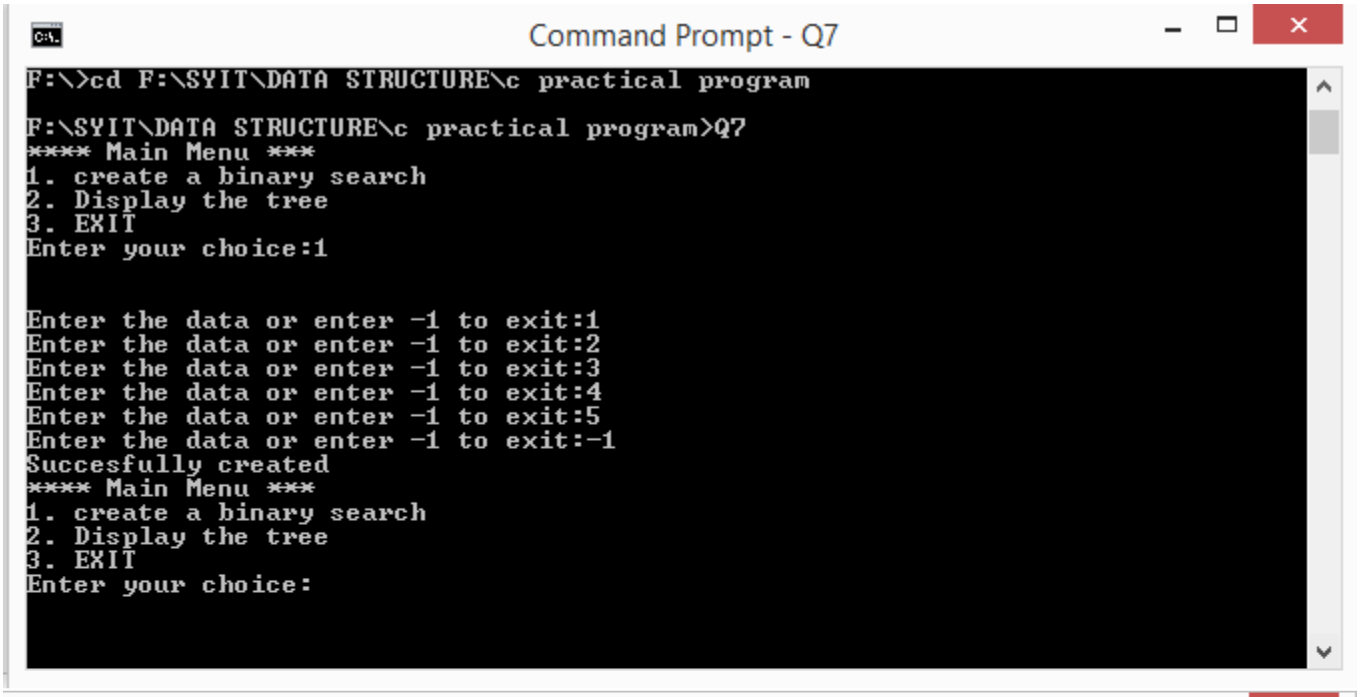
```
printf("\n\n");
switch(choice)
{
case 1:root=create(root);
break;
case 2:root=display(root);
break;
case 3:break;
}
}while(choice!=3);
return 0;
}
struct node *create(struct node *root)
{
struct node *newnode=NULL,*temp=NULL,*parent=NULL;
int val;
printf("Enter the data or enter -1 to exit:");
scanf("%d",&val);
while(val!=-1)
{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=val;
if(root==NULL)
{
root=newnode;
newnode->left=NULL;
newnode->right=NULL;
}
else
{
temp=root;
while(temp!=NULL)
```

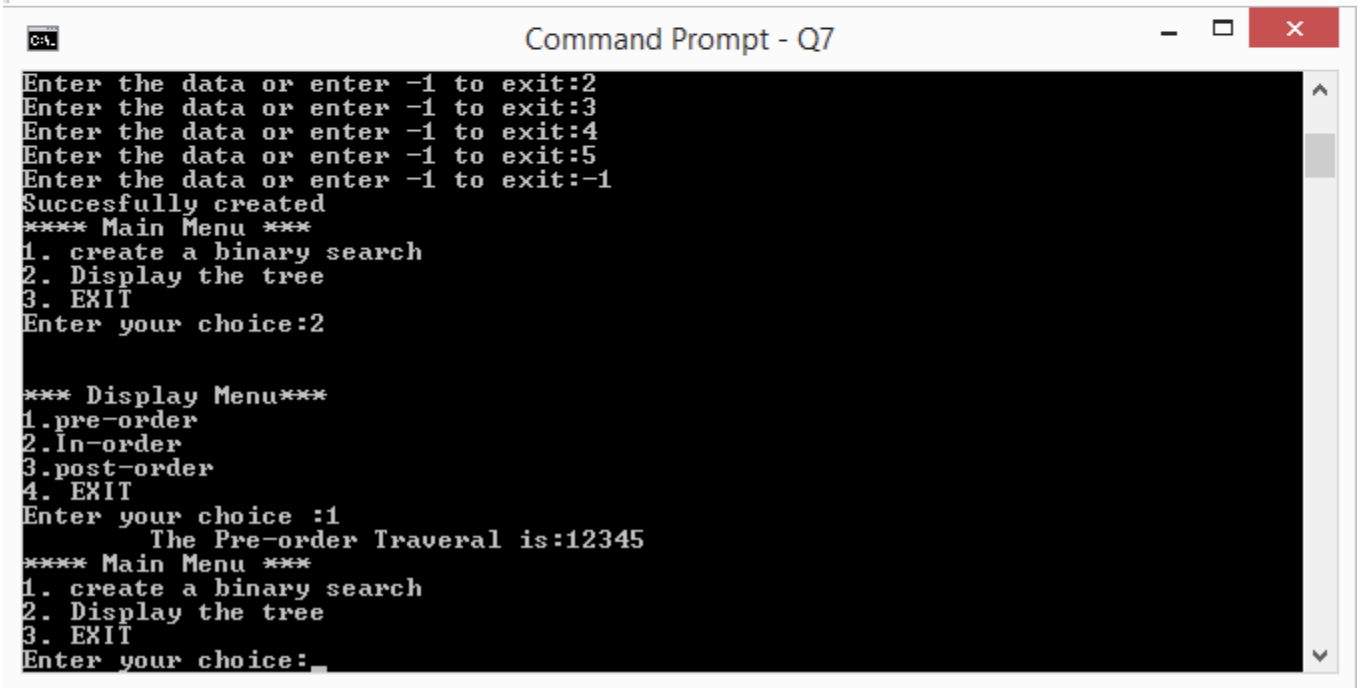
```
{
parent=temp;
if(val<temp->data)
{
temp=temp->left;
}
else
{
temp=temp->right;
}
}
if(val<parent->data)
{
parent->left=newnode;
newnode->left=NULL;
newnode->right=NULL;
}
else
{
parent->right=newnode;
newnode->left=NULL;
newnode->right=NULL;
}
}
printf("Enter the data or enter -1 to exit:");
scanf("%d",&val);
}
printf("Succesfully created \n");
return root;
}
struct node *display(struct node *root)
{
```

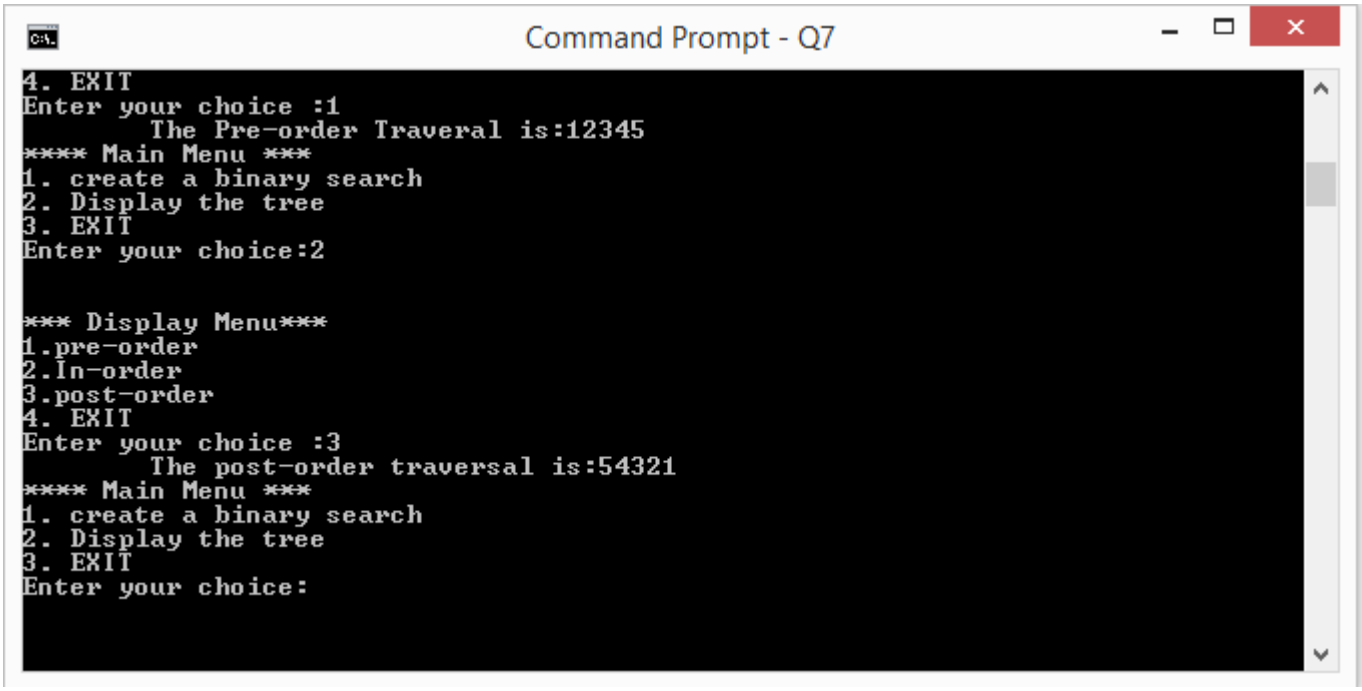
```
int choice1;
printf("*** Display Menu***\n");
printf("1.pre-order\n");
printf("2.In-order\n");
printf("3.post-order\n");
printf("4. EXIT\n");
printf("Enter your choice :");
scanf("%d",&choice1);
switch(choice1)
{
case 1:printf("\tThe Pre-order Traveral is:");
preorder(root);
break;
case 2:printf("\tThe in order traversal is:");
inorder(root);
break;
case 3:printf("\tThe post-order traversal is:");
postorder(root);
break;
case 4:break;
}
printf("\n");
return root;
}
void preorder(struct node *temp)
{
if(temp!=NULL)
{
printf("%d",temp->data);
preorder(temp->left);
preorder(temp->right);
}
```

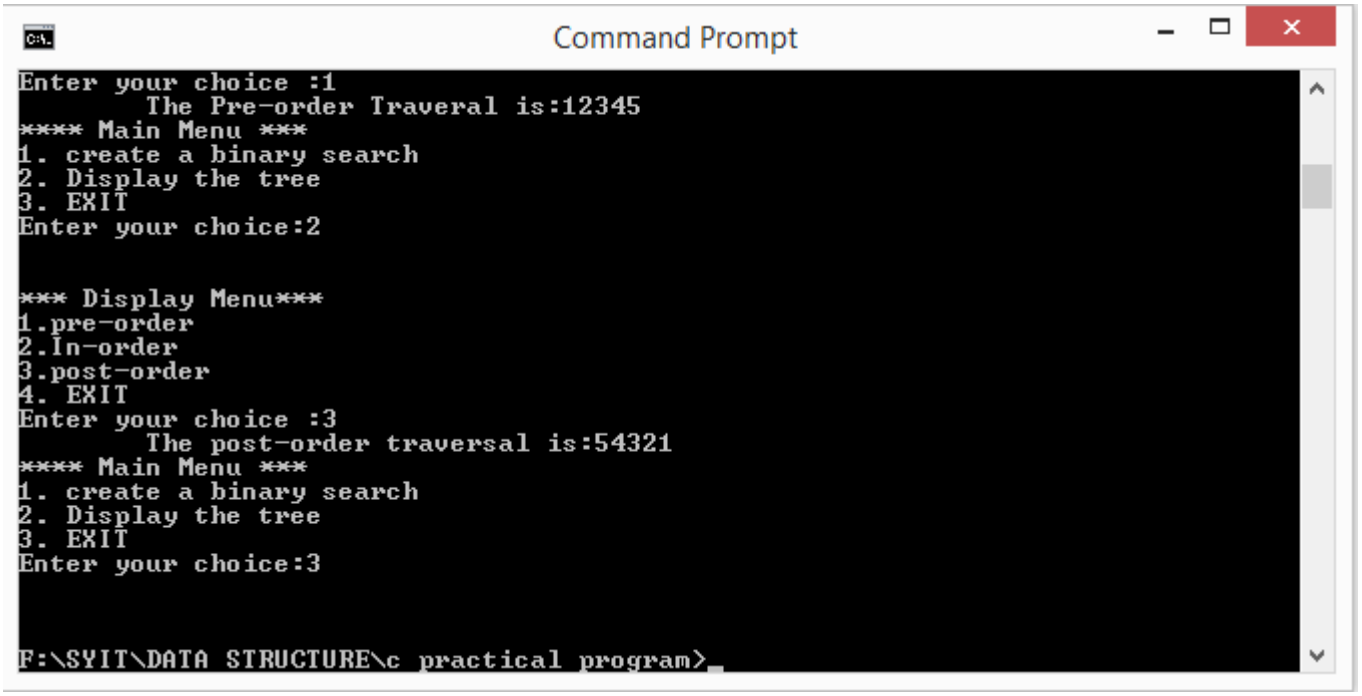
```
}
void postorder(struct node *temp)
{
if(temp!=NULL)
{
postorder(temp->left);
postorder(temp->right);
printf("%d",temp->data);

}
}
void inorder(struct node *temp)
{
if(temp!=NULL)
{
inorder(temp->left);
printf("%d",temp->data);
inorder(temp->right);
}
}
```

OUTPUT:









Implement the following data structure techniques:

PRACTICAL 8

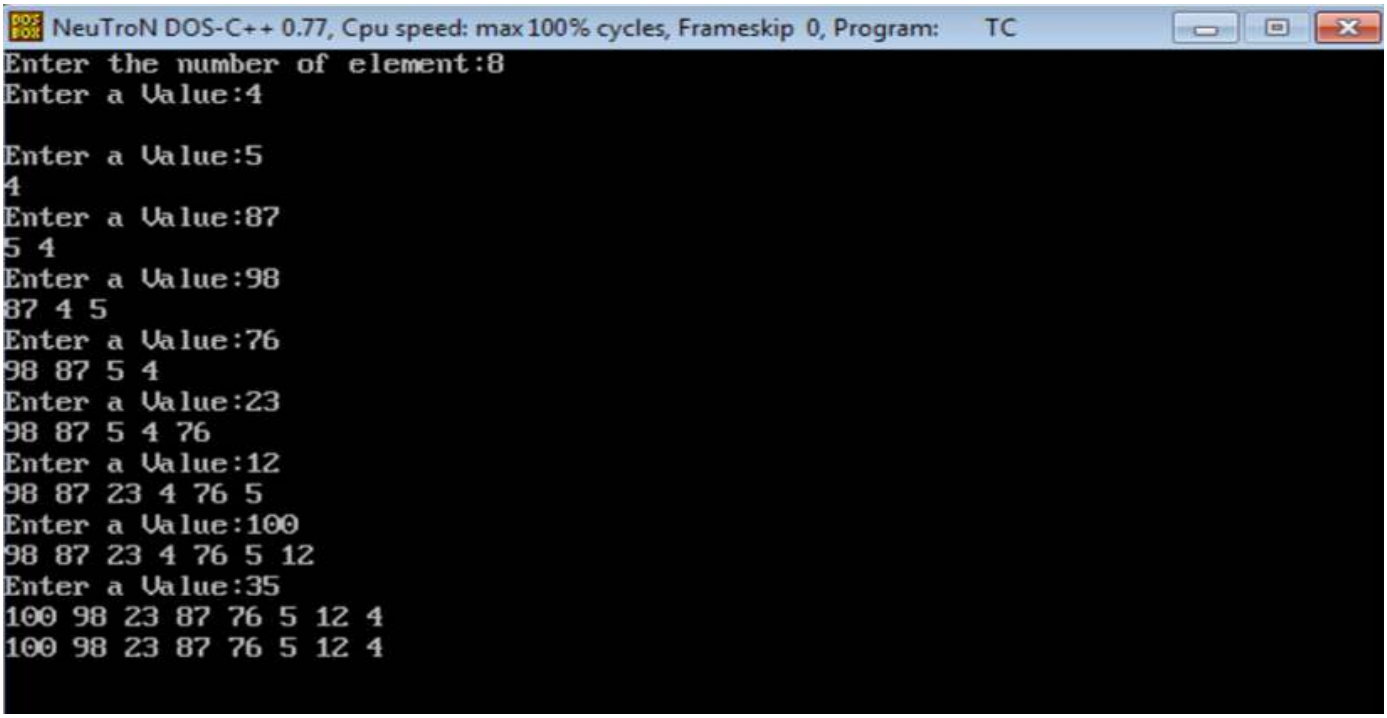
A . Write a program to insert the element into maximum heap.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define SIZE 30
int a[SIZE],n;
void maxHeapify(int a[],int i,int n1);
void buildHeap(int a[],int n1);
void heap_sort(int a[]);
void swap(int i,int j);
int length(int a[]);
void main()
{
int i,j;
clrscr();
printf("Enter the number of element:");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
printf("Enter a Value:");
scanf("%d",&a[i]);
buildHeap(a,i);
}
for(j=0;j<n;j++)
{
printf("%d ",a[j]);
}
printf("\n");
getch();
}
void buildHeap(int a[],int n1)
{
int i,j;
for(i=(n1/2)-1;i>=0;i--)
{
maxHeapify(a,i,n1);
}
for(j=0;j<n1;j++)
{
printf("%d ",a[j]);
}

printf("\n");
}

void maxHeapify(int a[],int i,int n1)
{
```

```
int max,l,r;
max=i;
l=2*i+1;
r=2*i+2;
if(l<n1 && r<n1)
{
if(a[l]>a[max])
{
max=l;
}
if(a[r]>a[max])
{
max=r;
}
}
else if(l<n1 && r>=n1)
{
if(a[l]>a[max])
{
max=l;
}
}
else if(l>=n1 && r<n1)
{
if(a[r]>a[max])
{
max=r;
}
}
if(i!=max)
{
swap(i,max);
maxHeapify(a,max,n1);
}
}
void swap(int i,int j)
{
int temp=a[i];
a[i]=a[j];
a[j]=temp;
}

int length(int a[])
{
int i=0;

while(a[i]!='\0')
{
i++;
}
return i;
}
```

OUTPUT:



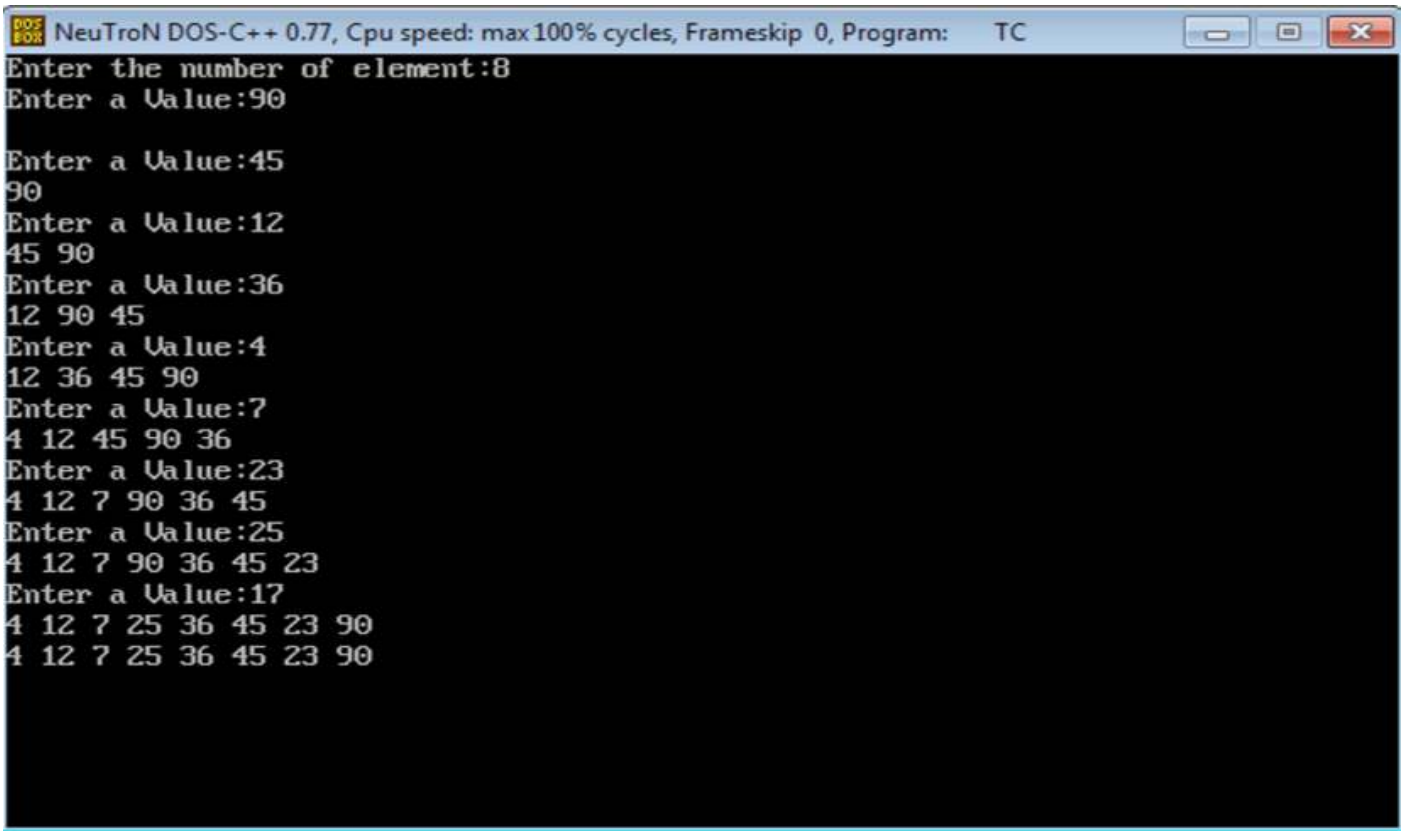
B. Write a program to insert the element into minimum heap.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define SIZE 30
int a[SIZE],n;
void minHeapify(int a[],int i,int n1);
void buildHeap(int a[],int n1);
void heap_sort(int a[]);
void swap(int i,int j);
int length(int a[]);
void main()
{
int i,j;
clrscr();
printf("Enter the number of element:");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
printf("Enter a Value:");
scanf("%d",&a[i]);
buildHeap(a,i);
}
for(j=0;j<n;j++)
{
printf("%d ",a[j]);
}
printf("\n");
getch();
}
void buildHeap(int a[],int n1)
{
int i,j;
for(i=(n1/2)-1;i>=0;i--)
{
minHeapify(a,i,n1);
}
for(j=0;j<n1;j++)
{
printf("%d ",a[j]);
}
printf("\n");

}
void minHeapify(int a[],int i,int n1)
{
int min,l,r;
min=i;
l=2*i+1;
```

```
r=2*i+2;
if(l<n1 && r<n1)
{
if(a[l]<a[min])
{
min=l;
}
if(a[r]<a[min])
{
min=r;
}
}
else if(l<n1 && r>=n1)
{
if(a[l]<a[min])
{
min=l;
}
}
else if(l>=n1 && r<n1)
{
if(a[r]<a[min])
{
min=r;
}
}
if(i!=min)
{
swap(i,min);
minHeapify(a,min,n1);
}
}

void swap(int i,int j)
{
int temp=a[i];
a[i]=a[j];
a[j]=temp;
}
int length(int a[])
{
int i=0;
while(a[i]!='\0')
{
i++;
}
return i;
}
```

OUTPUT:



Implement the following data structure techniques:

PRACTICAL 9

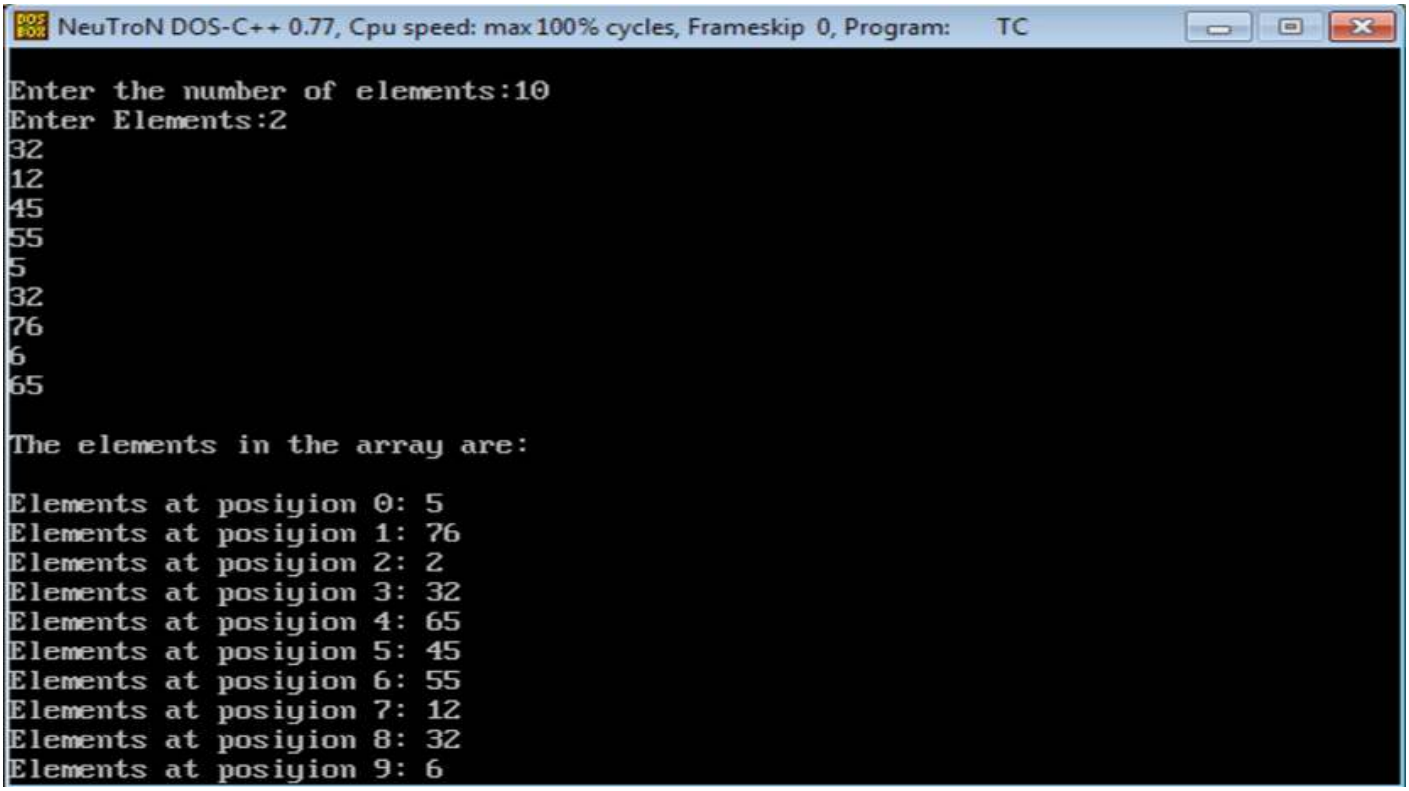
A. Write a program to implement the collision technique.

INPUT:

```
#include<stdio.h>
#include<conio.h>
#define size 10
int hash[20];
int arr[size];
int hasht(int key)
{
int i;
i=key % size;
return i;
}
int rehashq(int key,int j)
{
int i;
i=(key+(j*j))%size;
return i;
}
void main()
{
int key,i,n,s,op,j,k;
clrscr();
printf("\nEnter the number of elements:");
scanf("%d",&n);
for(i=0;i<size;i++)
hash[i]=-1;
printf("Enter Elements:");
for(i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<size;i++)
hash[i]=-1;
for(k=0;k<n;k++)
{
j=1;
key=arr[k];
i=hasht(key);
while(hash[i]!=-1)
{
i=rehashq(i,j);
j++;

}
hash[i]=key;
}
```

```
printf("\nThe elements in the array are:\n");
for(i=0;i<size;i++)
{
printf("\nElements at posiyion %d: %d",i,hash[i]);
}
getch();
}
```

OUTPUT:



B. Write a program to implement the concept of linear probing.

INPUT:

```
#include<stdio.h>
#define SIZE 10
#define FALSE 0
#define TRUE 1
#define h(x) x%SIZE
int data[SIZE],flag[SIZE],chain[SIZE];
void insert(int x);
int search(int x);
void print();
void deleteeL(int x);
void main()
{
int i,op,loc,x;
clrscr();
for(i=0;i<SIZE;i++)
{
flag[i]=FALSE;
chain[i]=-1;
}
do
{
printf("\n\n1-Insert\n2-Search\n3-Delete\n4-Print\n5-Quit");
printf("\nEnter your choice:");
scanf("%d",&op);
switch(op)
{
case 1:
{
printf("\nEnter a number to be insert:");
scanf("%d",&x);
insert(x);
break;
}
case 2:
{
printf("Enter the number to be searched:");
scanf("%d",&x);
if((loc=search(x))==-1)
{

printf("\nElements not found");
}
else
{
printf("\n***Found at location=%d",loc);
}
break;
}
```

```
case 3:
{
printf("Enter the value to be delete:");
scanf("%d",&x);
deleteeL(x);
break;
}
case 4:
{
print();
break;
}
case 5:
{
exit(0);
break;
}
default:
{
printf("\nInvalid Choice..");
}
}
}while(op!=5);
getch();
}
void insert(int x)
{
int i=0,j,start;
start=h(x);
if(flag[start]==0)
{
data[start]=x;
flag[start]=1;
return;

}
i=0;
j=start;
while(flag[j]&&i<SIZE)
{
j=(j+1)%SIZE;
i++;
}
if(i==SIZE)
{
printf("\nTable is full..");
return;
}
data[j]=x;
flag[j]=1;
chain[j]=-1;
i=start;
while(chain[i]!=-1)
```

```
{
i=chain[i];
chain[i]=j;
}
}
int search(int x)
{
int i=0,j;
j=h(x);
while((i<SIZE) && (flag[j]) && (data[j]%SIZE) !=(x%SIZE))
{
i++;
j=(j+1)%SIZE;
}
if((!flag[j])||(i==SIZE))
{
return(-1);
}
while(j!=-1)
{
if(data[j]==x)
return(j);

j=chain[j];
}
return(-1);
}
void deleteeL(int x)
{
int i=0,j=0,loc;
loc=search(x);
if(loc==-1)
{
printf("Elements not present");
}
else
{
data[loc]=0;
flag[loc]=FALSE;
printf("Values deleted!!");
while(chain[i]!=loc)
{
i++;
}
chain[i]=chain[loc];
chain[loc]=-1;
}
}
void print()
{
int i;
for(i=0;i<SIZE;i++)
{
```

```
if(flag[i])
{
printf("\n(%d)%d %d",i,data[i],chain[i]);
}
else
{
printf("\n(%d)--- %d",i,chain[i]);
}
}
}
```

OUTPUT:

Implement the following data structure techniques:

PRACTICAL 10

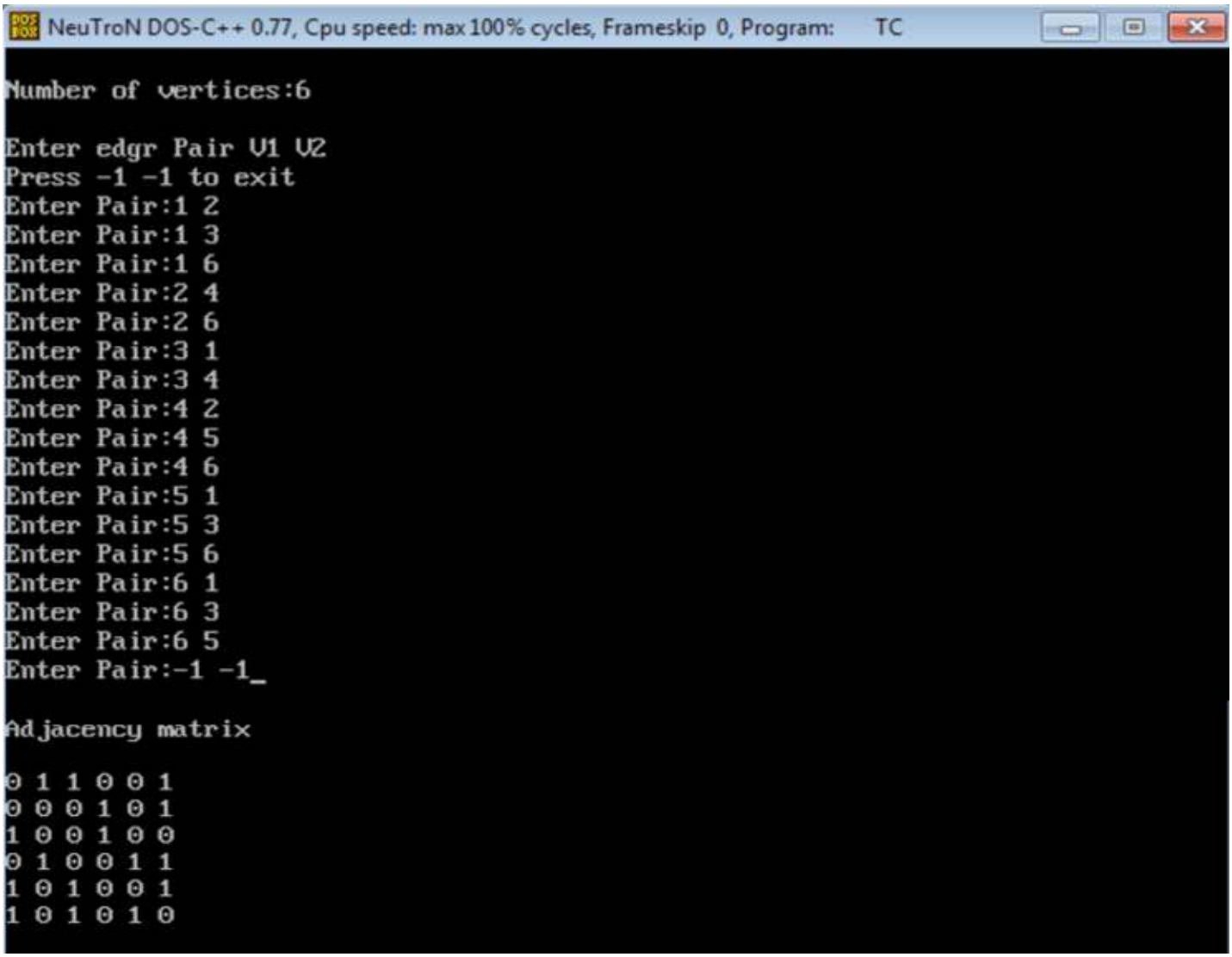
A. Write a program to generate the adjacency matrix.

INPUT:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int **adjfmatrix;
int r,c,v;
clrscr();
printf("\nNumber of vertices:");
scanf("%d",&v);
adjfmatrix=(int **)malloc(sizeof(int **)*v);
for(r=0;r<v;r++)
{
adjfmatrix[r]=(int *)malloc(sizeof(int)*v);
}
for(r=0;r<v;r++)
{
for(c=0;c<v;c++)
{
adjfmatrix[r][c]=0;
}
}
r=0;
c=0;
printf("\nEnter edgr Pair V1 V2\n");
printf("Press -1 -1 to exit\n");
do
{
printf("Enter Pair:");
scanf("%d %d",&r,&c);
if(r>0 && r<=v && c>0 && c<=v)
{
adjfmatrix[r-1][c-1]=1;
}
}while(r>0 && c>0);
printf("\nAdjacency matrix\n");
printf(" ");
printf("\n");

for(r=0;r<v;r++)
{
for(c=0;c<v;c++)
{
printf("%d ",adjfmatrix[r][c]);
}
printf("\n");
```

```
 }
 getch();
}
```

OUTPUT:



B. Write a program for shortest path diagram.

INPUT:

```
#include<stdio.h>
#define INF 999
#define SIZE 10
void dijkstra(int A[SIZE][SIZE],int n,int s);
void main()
{
int A[SIZE][SIZE],i,j,n,u;
clrscr();
printf("Enter no of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&A[i][j]);
}
}
printf("\nEnter the string node:");
scanf("%d",&u);
dijkstra(A,n,u);
getch();
}
void dijkstra(int A[SIZE][SIZE],int n,int s)
{
int cost[SIZE][SIZE],dist[SIZE],parent[SIZE];
int visited[SIZE],count,mindist,next,i,j;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(A[i][j]==0)
{
cost[i][j]=INF;
}
else
{
cost[i][j]=A[i][j];
}
}
}

for(i=0;i<n;i++)
{
dist[i]=cost[s][i];
parent[i]=s;
visited[i]=0;
}
```

```
 dist[s]=0;
 visited[s]=1;
 count=1;
 while(count<n-1)
 {
 mindist=INF;
 for(i=0;i<n;i++)
 {
 if(dist[i]<mindist&&!visited[i])
 {
 mindist=dist[i];
 next=i;
 }
 }
 visited[next]=1;
 for(i=0;i<n;i++)
 {
 if(!visited[i])
 if(mindist+cost[next][i]<dist[i])
 {
 dist[i]=mindist+cost[next][i];
 parent[i]=next;
 }
 }
 count++;
 }
 for(i=0;i<n;i++)
 {
 if(i!=s)
 {
 printf("\nDist of node%d=%d",i,dist[i]);
 printf("\nPath=%d",i);
 j=i;
 do
 {
 j=parent[j];

 printf("<

 -%d",j);

 }while(j!=s);
 printf("
 \n");

 }
 }
}
```

OUTPUT: