# MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

# DATA STRUCTURE LAB

# Practical file

SUBMITTED BY:

ASMITA JAIN

0901EO201017

SUBMITTED TO:

PROF. NAMRATA AGRAWAL

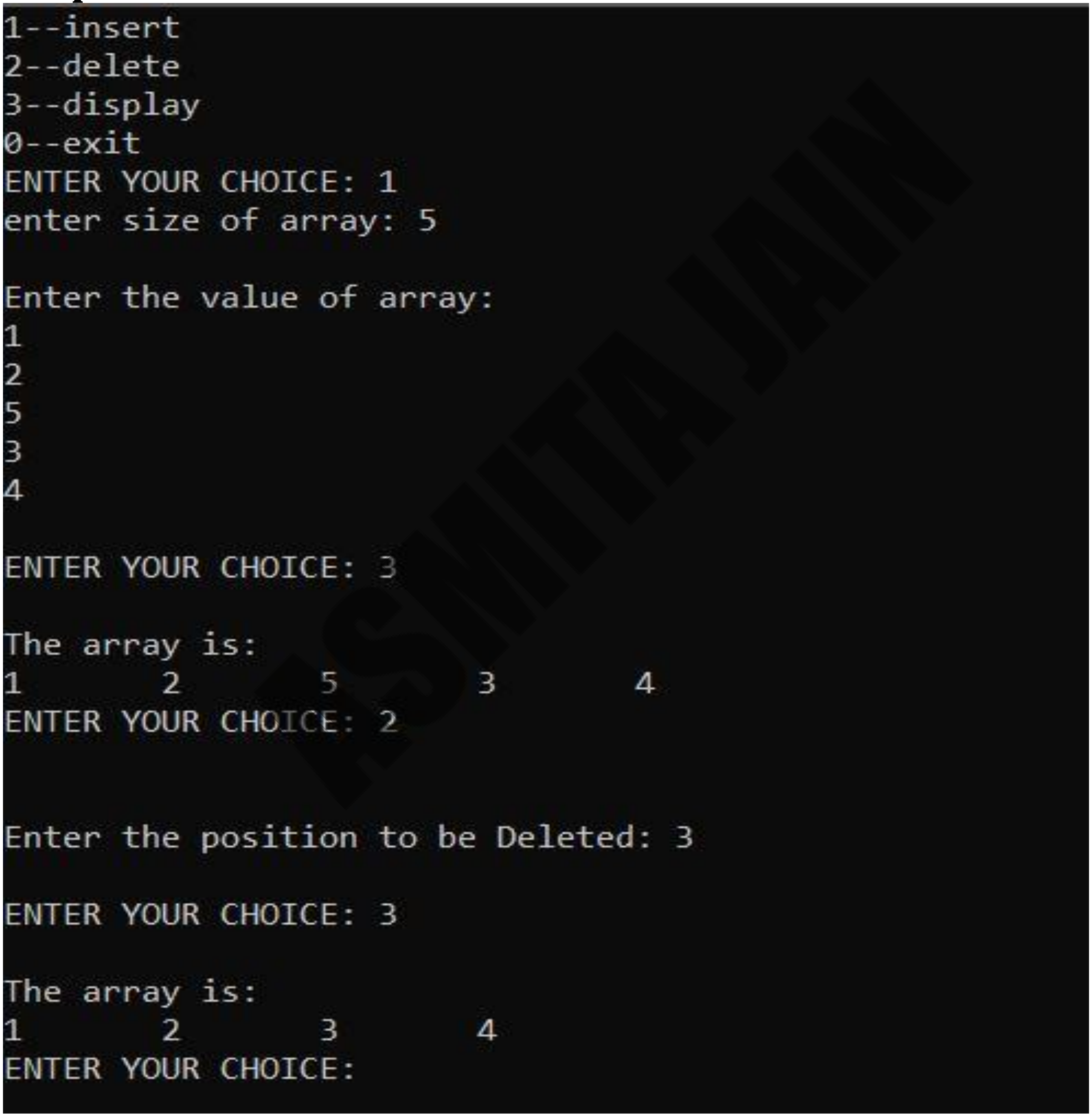
# Index

**1. Write a program to implement all operations (create, insertion and deletion, display) in array.**

**Code:**

```
#include <iostream>
#define MAX 100

using namespace std;
void insert(int a[]);
void display(int a[]);
void Delete(int a[], int p);
int size;
int main()
{
  int choice;
  int a[MAX], n, p;
  cout<<"1--insert\n2--delete\n3--display\n0--exit";
  while(1)
  {
    cout<<"\nENTER YOUR CHOICE: ";
    cin>>choice;
    switch(choice)
    {
    case 1:
     insert(a);
     break;
    case 2:
     cout << "\n\nEnter the position to be Deleted: ";
     cin >> p;
     p--;
     Delete(a,p);
     break;
    case 3:
        display(a);
        break;
       case 0:
        exit(0);
       default:
        cout<<"\nWRONG CHOICE";
       }

    }
    return 0;
}
void insert(int a[])
{
    cout<<"enter size of array: ";
    cin>>size;
    cout << "\nEnter the value of array: \n";
    for(int i=0; i<size; ++i)
    {
        cin >> a[i];
    }
}

void display(int a[])
{
    cout << "\nThe array is: \n";
    for(int i=0; i<size; ++i)
    {
        cout <<a[i] ;
        cout << "\t";
    }
}
void Delete(int a[], int pos)
{
    for(int i=pos; i<size; ++i)
    {
        a[i] = a[i+1];
    }
    size--;
}
```

**Output:**



```
1--insert
2--delete
3--display
0--exit
ENTER YOUR CHOICE: 1
enter size of array: 5

Enter the value of array:
1
2
5
3
4

ENTER YOUR CHOICE: 3

The array is:
1       2       5       3       4
ENTER YOUR CHOICE: 2


Enter the position to be Deleted: 3

ENTER YOUR CHOICE: 3

The array is:
1       2       3       4
ENTER YOUR CHOICE:
```

## 2. Write a program to implement all operations (create, insertion and deletion, display) in singly linked list

**Code:**

```
#include <iostream>
#include <malloc.h>
#include <process.h>
using namespace std;
struct node
{
    int a;
    node *next;
} *head = NULL;
void insert(int newdata)
{

    node p = (node)malloc(sizeof(node));
    p->a = newdata;
    p->next = head;
    head = p;
}

void display()
{
    node *item;
    item = head;
    cout << "\nLIST :\n";
    while (item != NULL)
    {
        cout << "node item : " << item->a << endl;
        item = item->next;
```

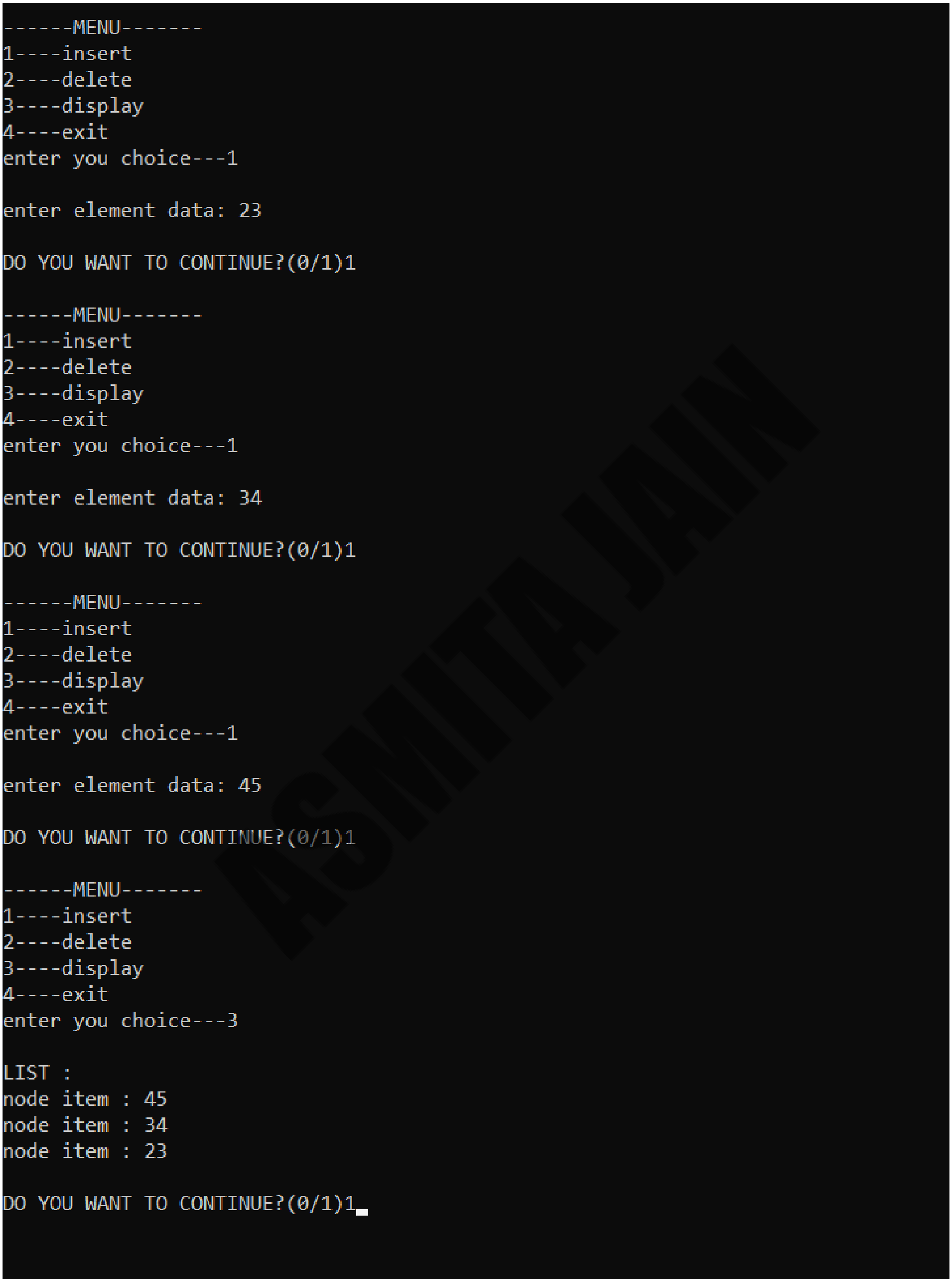
```
   }
}
void delete_(int key)
{
   int flag = 0;
   if (head == NULL)
      cout << "\no element";
   else
   {
      node *temp;
      if (head->a == key)
      {
         temp = head;
         head = head->next;
         cout << "\nNode deleted ";
         free(temp);
      }
      else
      {
         node *item;
         item = head;
         while (item->next != NULL)
         {
            if (item->next->a == key)
            {
               temp = item->next;
               item->next = item->next->next;
               free(temp);
               flag = 1;
               cout << "\nnode deleted!";
               break;
```

```
                }
                else
                    item = item->next;
            }
            if (flag == 0)
                cout << "\nNO such element";
        }
    }
}

int main()
{
    int ch = 1, data, choice, key;

    while (ch)
    {
        cout << "\n------MENU-------" << endl;
        cout << "1----insert" << endl;
        cout << "2----delete" << endl;
        cout << "3----display" << endl;
        cout << "4----exit" << endl;
        cout << "enter you choice---";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "\nenter element data: ";
            cin >> data;
            insert(data);
            break;
        case 2:
```

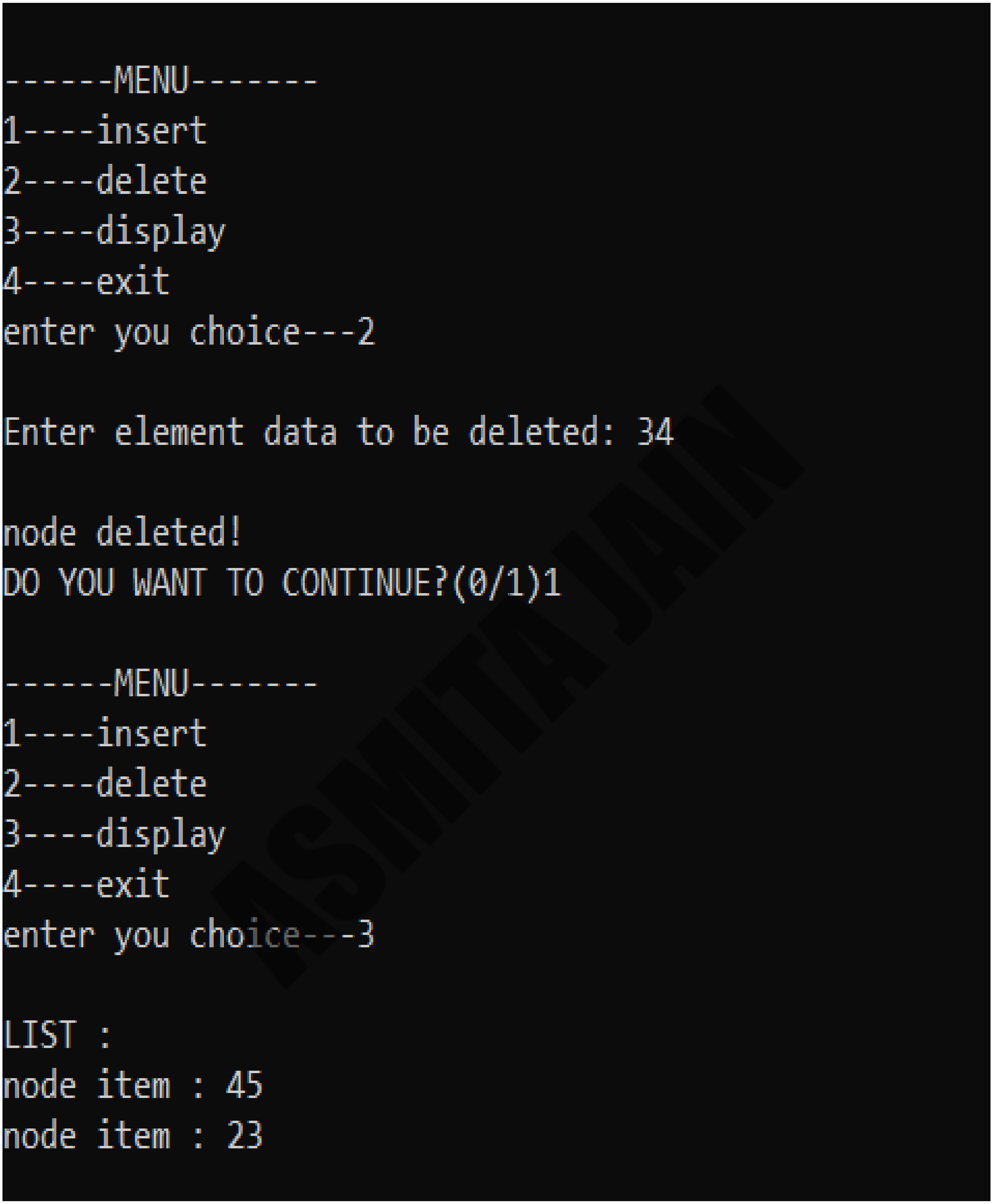
```
      cout << "\nEnter element data to be deleted: ";
      cin >> key;
      delete_(key);
      break;
    case 3:
      display();
      break;
    case 4:
      exit(0);
    default:
      cout << "\nWrong choice entered";
    }
    cout << "\nDO YOU WANT TO CONTINUE?(0/1)";
    cin >> ch;
  }
  return 0;
}
```

**(output on next PAGE)**

## Output:



```
------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---1

enter element data: 23

DO YOU WANT TO CONTINUE?(0/1)1

------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---1

enter element data: 34

DO YOU WANT TO CONTINUE?(0/1)1

------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---1

enter element data: 45

DO YOU WANT TO CONTINUE?(0/1)1

------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---3

LIST :
node item : 45
node item : 34
node item : 23

DO YOU WANT TO CONTINUE?(0/1)1
```



```
------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---2

Enter element data to be deleted: 34

node deleted!
DO YOU WANT TO CONTINUE?(0/1)1

------MENU-------
1----insert
2----delete
3----display
4----exit
enter you choice---3

LIST :
node item : 45
node item : 23
```

## 3.Write a program to perform insertion, deletion and traversing in doubly linked list

**Code:**

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

struct node
{
   struct node *prev;
   int ddata;
   struct node *next;
}*ptr=NULL,*temp=NULL,*temp1=NULL,*temp2,*temp4;

void insert_beg();
void insert_end();
void insert_i();
void Display_beg();
void Display_end(int);
void Delete();
void new_node();
int count=0;
void main()
{
   int ch;


   printf("\n1 - Insert at beginning");
   printf("\n2 - Insert at end");
   printf("\n3 - Insert at position i");
   printf("\n4 - Delete at position i");
printf("\n5 - Display from beginning");
printf("\n6 - Display from end");
printf("\n0 - Exit");

while (1)
{
    printf("\n Enter choice : ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
        insert_beg();
        break;
    case 2:
        insert_end();
        break;
    case 3:
        insert_i();
        break;
    case 4:
        Delete();
        break;
    case 5:
        Display_beg();
        break;
    case 6:
        temp2 = ptr;
        if (temp2 == NULL)
            printf("\nError : List empty to display ");
        else
        {
            printf("\nReverse order of linked list is : ");
            Display_end(temp2->ddata);
         }
         break;
      case 0:
         exit(0);
      default:
         printf("\nWrong choice ");
      }
   }
}


void new_node()
{
   int data;

   temp =(struct node *)malloc(1*sizeof(struct node));
   temp->prev = NULL;
   temp->next = NULL;
   printf("\nEnter value to node : ");
   scanf("%d", &data);
   temp->ddata = data;
   count++;
}


void insert_beg()
{
   if (ptr == NULL)
   {
        new_node();
        ptr = temp;
        temp1 = ptr;
    }
    else
    {
        new_node();
        temp->next = ptr;
        ptr->prev = temp;
        ptr = temp;
    }
}


void insert_end()
{
    if (ptr == NULL)
    {
        new_node();
        ptr = temp;
        temp1 = ptr;
    }
    else
    {
        new_node();
        temp1->next = temp;
        temp->prev = temp1;
        temp1 = temp;
    }
}
```

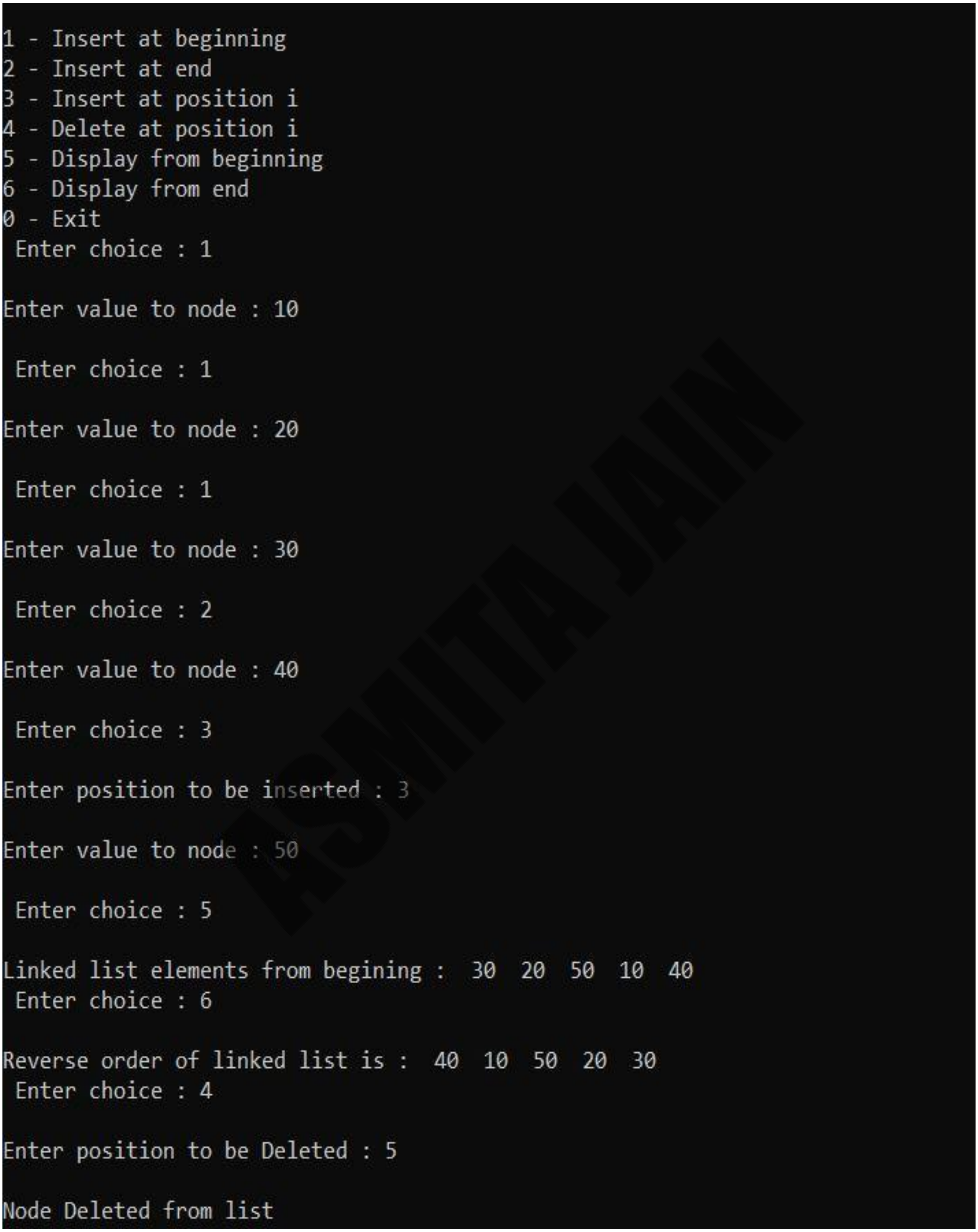
```
void insert_i()
{
    int pos, i = 2;

    printf("\nEnter position to be inserted : ");
    scanf("%d", &pos);
    temp2 = ptr;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\nPosition out of range to insert");
        return;
    }
    if ((ptr == NULL) && (pos != 1))
    {
        printf("\nEmpty list cannot insert other than 1st position");
        return;
    }
    if ((ptr == NULL) && (pos == 1))
    {
        new_node();
        ptr = temp;
        temp1 = ptr;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
```

```
         i++;
      }
      new_node();
      temp->prev = temp2;
      temp->next = temp2->next;
      temp2->next->prev = temp;
      temp2->next = temp;
   }
}

void Delete()
{
   int i = 1, pos;

   printf("\nEnter position to be Deleted : ");
   scanf("%d", &pos);
   temp2 = ptr;

   if ((pos < 1) || (pos >= count + 1))
   {
      printf("\nError : Position out of range to Delete");
      return;
   }
   if (ptr == NULL)
   {
      printf("\nError : Empty list no elements to Delete");
      return;
   }
   else
   {
      while (i < pos)
```

```
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf("\nNode Deleted from list");
                free(temp2);
                temp2 = ptr = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf("\nNode Deleted from list");
            return;
        }
        temp2->next->prev = temp2->prev;
        if (i != 1)
            temp2->prev->next = temp2->next;
        if (i == 1)
            ptr = temp2->next;
        printf("\nNode Deleted");
        free(temp2);
    }
    count--;
}
```

```
void Display_beg()
{
   temp2 = ptr;

   if (temp2 == NULL)
   {
      printf("List empty to display \n");
      return;
   }
   printf("\nLinked list elements from begining : ");

   while (temp2->next != NULL)
   {
      printf(" %d ", temp2->ddata);
      temp2 = temp2->next;
   }
   printf(" %d ", temp2->ddata);
}


void Display_end(int i)
{
   if (temp2 != NULL)
   {
      i = temp2->ddata;
      temp2 = temp2->next;
      Display_end(i);
      printf(" %d ", i);
   }
}
```

## OUTPUT:



```
1 - Insert at beginning
2 - Insert at end
3 - Insert at position i
4 - Delete at position i
5 - Display from beginning
6 - Display from end
0 - Exit
 Enter choice : 1

Enter value to node : 10

 Enter choice : 1

Enter value to node : 20

 Enter choice : 1

Enter value to node : 30

 Enter choice : 2

Enter value to node : 40

 Enter choice : 3

Enter position to be inserted : 3

Enter value to node : 50

 Enter choice : 5

Linked list elements from begining :  30  20  50  10  40
 Enter choice : 6

Reverse order of linked list is :  40  10  50  20  30
 Enter choice : 4

Enter position to be Deleted : 5

Node Deleted from list
```

## 4.i. Write a program to implement stack using array.

**Code:**

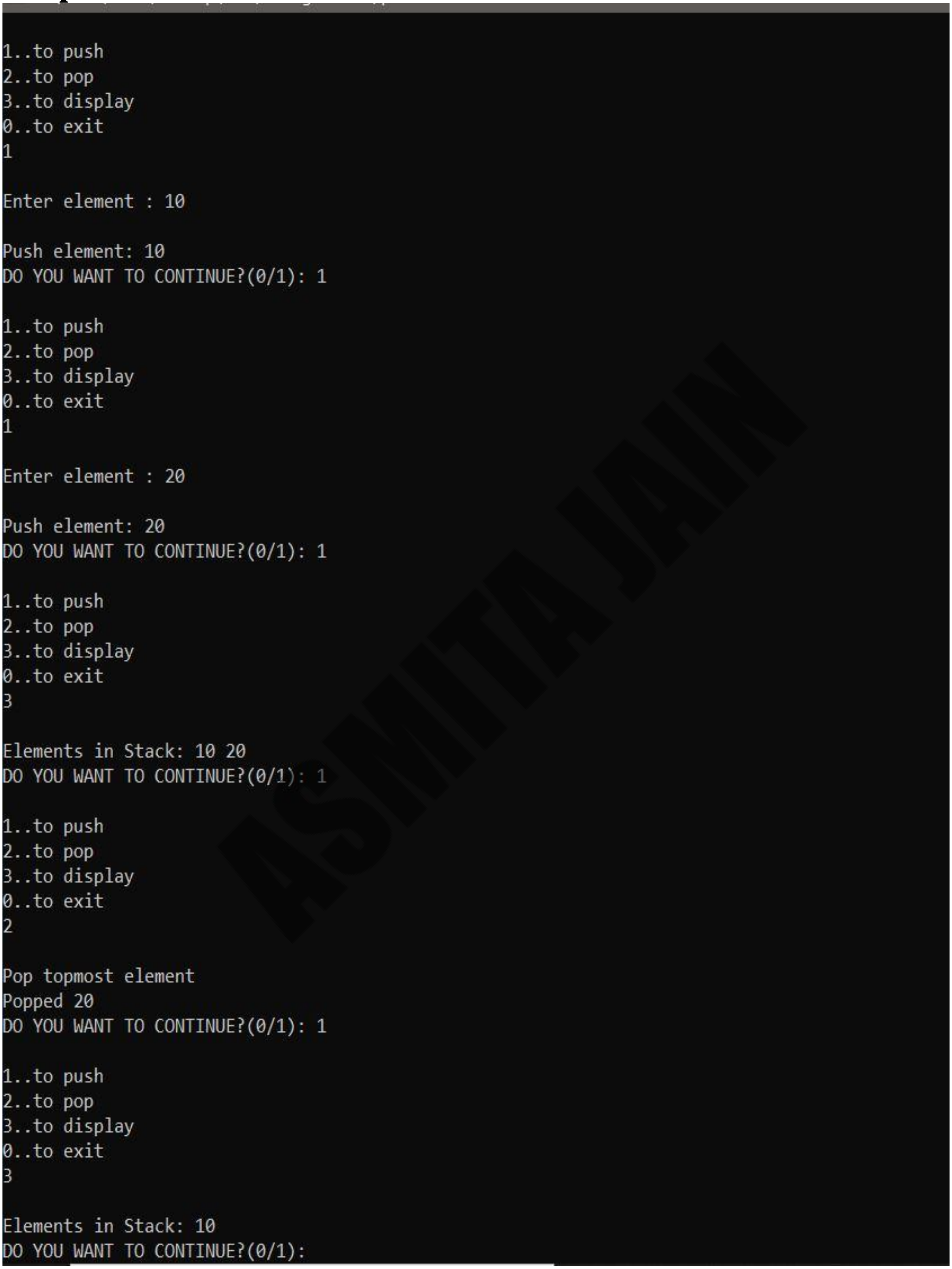
```
#include<stdio.h>
#include<process.h>
int nTop=-1;
int *p = NULL;

void push(int n)
{
 printf("\nPush element: %d", n);
 if(nTop>9)
 printf("Overflow");
 else
 {
  nTop++;
  p[nTop] = n;
 }
}

void pop()
{
 printf("\nPop topmost element");
 if(nTop<0)
 printf("\nUnderflow");
 else
 {
  printf("\nPopped %d",p[nTop]);
  p[nTop] = -1;
  nTop--;
  }
```

```
}

void DisplayStack()
{
 int i=0;
 if(nTop<0)
 printf("\nStack is empty");
 else
 {
  printf("\nElements in Stack: ");
  for(; i<=nTop;i++)
  printf("%d ", p[i]);
 }
}

int main()
{
  int ch=1,choice,x;
  p = (int *)malloc(sizeof(int)*10);

  while(ch)
  {
    printf("\n1..to push\n2..to pop\n3..to display\n0..to exit\n");
    scanf("%d",&choice);
    switch(choice)
    {
    case 1:
      printf("\nEnter element : ");
      scanf("%d",&x);
      push(x);
      break;
```

```
        case 2:
            pop();
            break;
        case 3:
            DisplayStack();
            break;
        case 0:
            exit(0);
        default:
            printf("\nWRONG CHOICE!");

        }
        printf("\nDO YOU WANT TO CONTINUE?(0/1): ");
        scanf("%d",&ch);

    }
    return 0;
}
```

**(OUTPUT ON NEXT PAGE)**

## Output:



```
1..to push
2..to pop
3..to display
0..to exit
1

Enter element : 10

Push element: 10
DO YOU WANT TO CONTINUE?(0/1): 1

1..to push
2..to pop
3..to display
0..to exit
1

Enter element : 20

Push element: 20
DO YOU WANT TO CONTINUE?(0/1): 1

1..to push
2..to pop
3..to display
0..to exit
3

Elements in Stack: 10 20
DO YOU WANT TO CONTINUE?(0/1): 1

1..to push
2..to pop
3..to display
0..to exit
2

Pop topmost element
Popped 20
DO YOU WANT TO CONTINUE?(0/1): 1

1..to push
2..to pop
3..to display
0..to exit
3

Elements in Stack: 10
DO YOU WANT TO CONTINUE?(0/1):
```

## 4.ii. Write a program to implement stack using linked list.

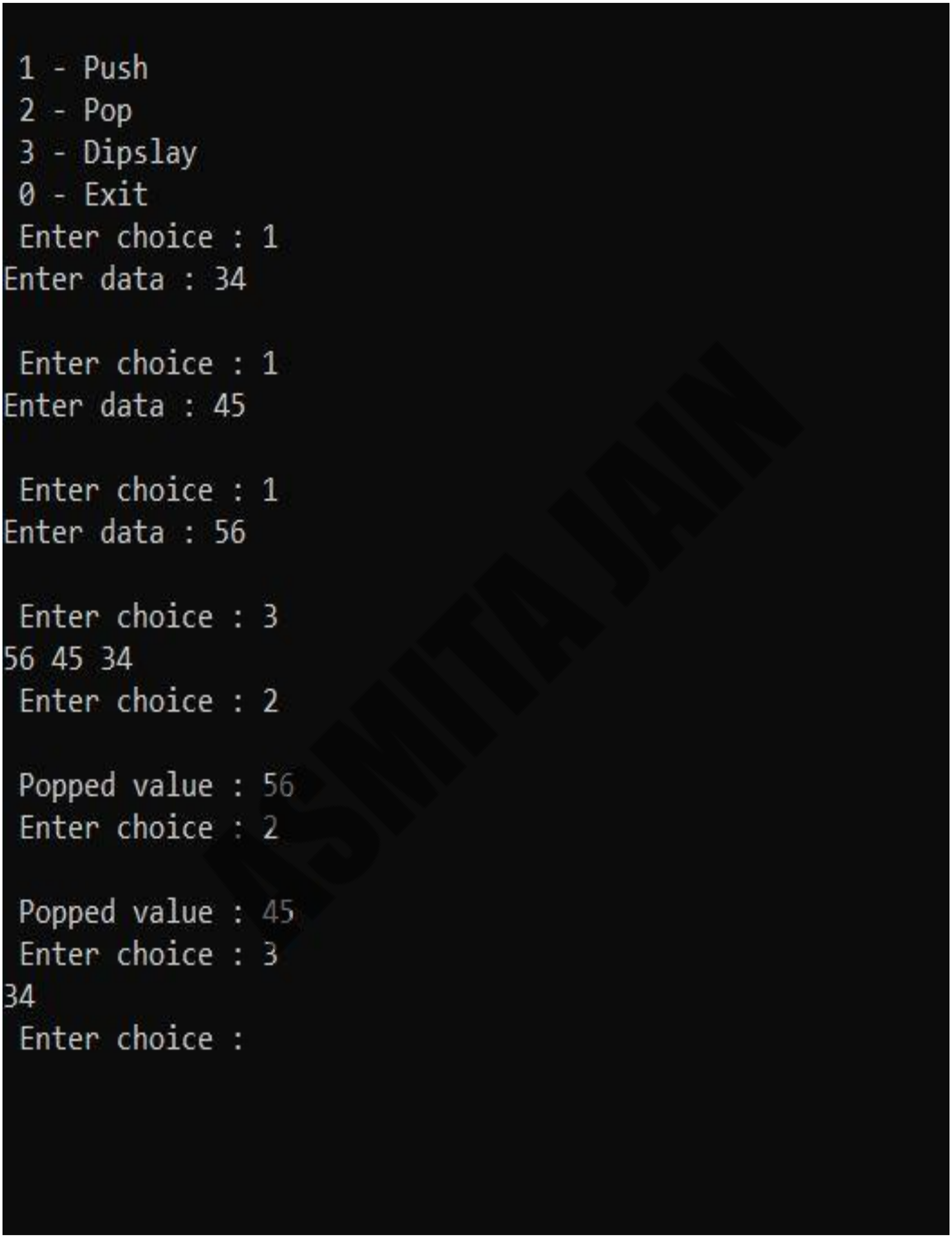
**Code:**

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

struct node
{
   int pdata;
   struct node *next;
}*top=NULL,*ptr,*temp;


void push(int data);
void pop();
int empty();
void display();


int main()
{
   int no, ch, e;

   printf("\n 1 - Push");
   printf("\n 2 - Pop");
   printf("\n 3 - Dipslay");
   printf("\n 0 - Exit");

   while (1)
   {
        printf("\n Enter choice : ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("Enter data : ");
            scanf("%d", &no);
            push(no);
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 0:
            exit(0);
        default :
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
    return 0;
}


void push(int data)
{
    if (empty())
    {
        top =(struct node *)malloc(1*sizeof(struct node));
        top->next = NULL;
        top->pdata = data;
    }
    else
    {
        temp =(struct node *)malloc(1*sizeof(struct node));
        temp->next = top;
        temp->pdata = data;
        top = temp;
    }
}


void display()
{
    ptr = top;

    if (empty())
    {
        printf("Stack is empty");
        return;
    }

    while (ptr != NULL)
    {
        printf("%d ", ptr->pdata);
        ptr = ptr->next;
    }
 }
```

```
void pop()
{
   ptr = top;

   if (empty())
   {
      printf("\n Error : Trying to pop from empty stack");
      return;
   }
   else
      ptr = ptr->next;
   printf("\n Popped value : %d", top->pdata);
   free(top);
   top = ptr;
}


int empty()
{
   if (top == NULL)
      return 1;
   else
      return 0;
}
```

**(OUTPUT ON NEXT PAGE)**

## Output:



```

 1 - Push
 2 - Pop
 3 - Dipslay
 0 - Exit
 Enter choice : 1
Enter data : 34

 Enter choice : 1
Enter data : 45

 Enter choice : 1
Enter data : 56

 Enter choice : 3
56 45 34
 Enter choice : 2

 Popped value : 56
 Enter choice : 2

 Popped value : 45
 Enter choice : 3
34
 Enter choice :
```

## 5.i Write a program to implement queue using array

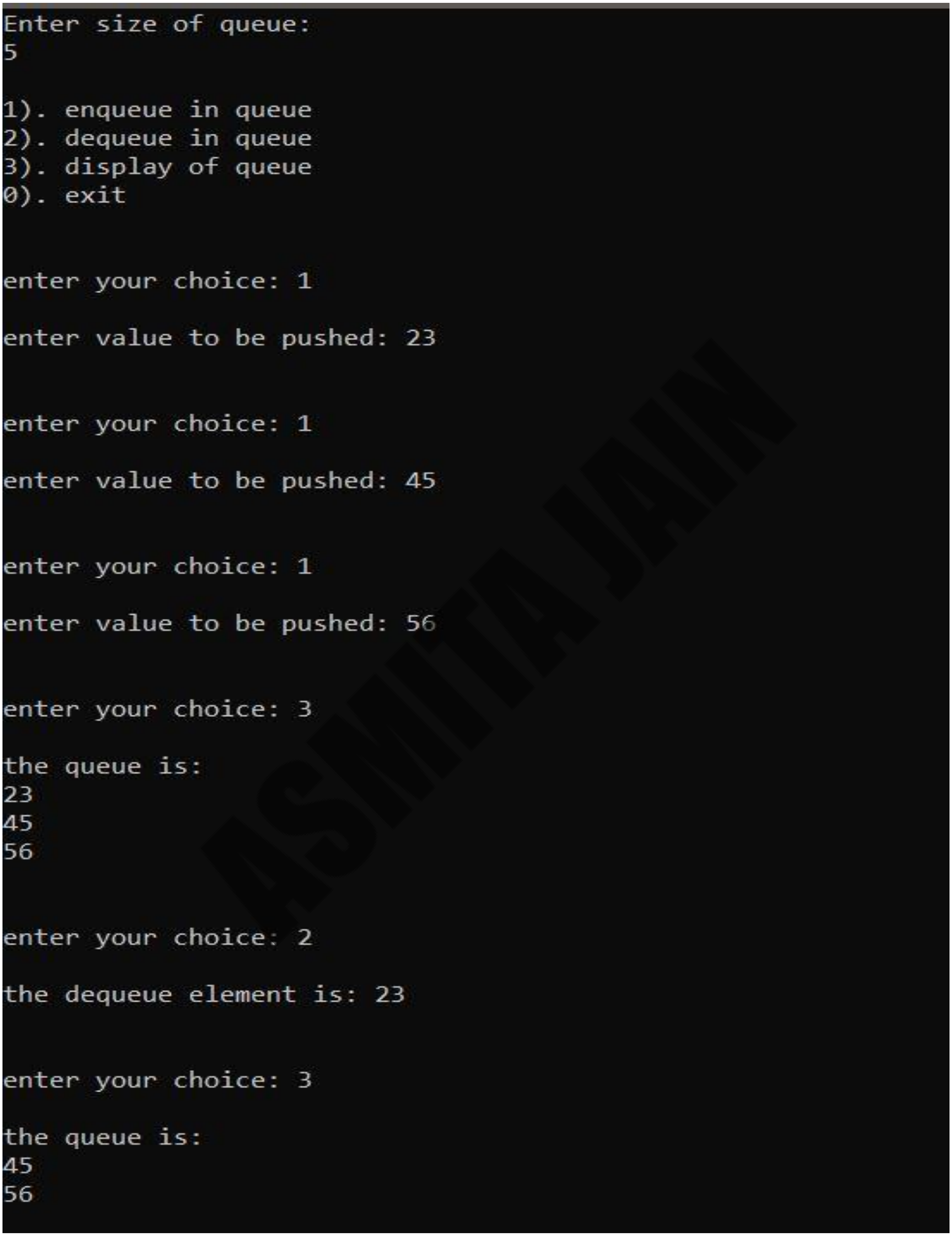
**Code:**

```
#include <iostream>
#include<process.h>
using namespace std;
int a[100], rear=-1, front = -1,size;
void enqueue();
void dequeue();
void display();
int is_full();
int is_empty_();

int main()
{
   cout<<"Enter size of queue: "<<endl;
   cin>>size;
   int ch, val;
   cout << "\n1). enqueue in queue"<< endl;
   cout << "2). dequeue in queue"<< endl;
   cout << "3). display of queue"<< endl;
   cout << "0). exit"<< endl;
   while(1)
   {
      cout << "\nenter your choice: ";
      cin >> ch;
      switch(ch)
      {
      case 1:
          enqueue();
          break;

      case 2:
          dequeue();
          break;

      case 3:
          display();
          break;
      case 0:

          exit(0);
      default:

          cout << "\ninvalid selection" << endl;
      }
   }
   return 0;
}
void enqueue()
{
   int val;
   if(is_full())
      cout<<"\nQUEUE Full";
   else
   {
      cout << "\nenter value to be pushed: ";
      cin >> val;
      rear++;
      a[rear]= val;
      if(front==-1)
        front++;
   }

}
void dequeue()
{
   if(is_empty_())
   {
      cout << "\nthere is no element for dequeue"<< endl;
   }
   else
   {    cout << "\nthe dequeue element is: " << a[front]<< endl;
      front++;
   }
}
void display()
{
   if(is_empty_())
   {
      cout << "there is no element for display";
   }
   else
   {   cout << "\nthe queue is:" << endl;
      for(int i=front; i<=rear; ++i)
```

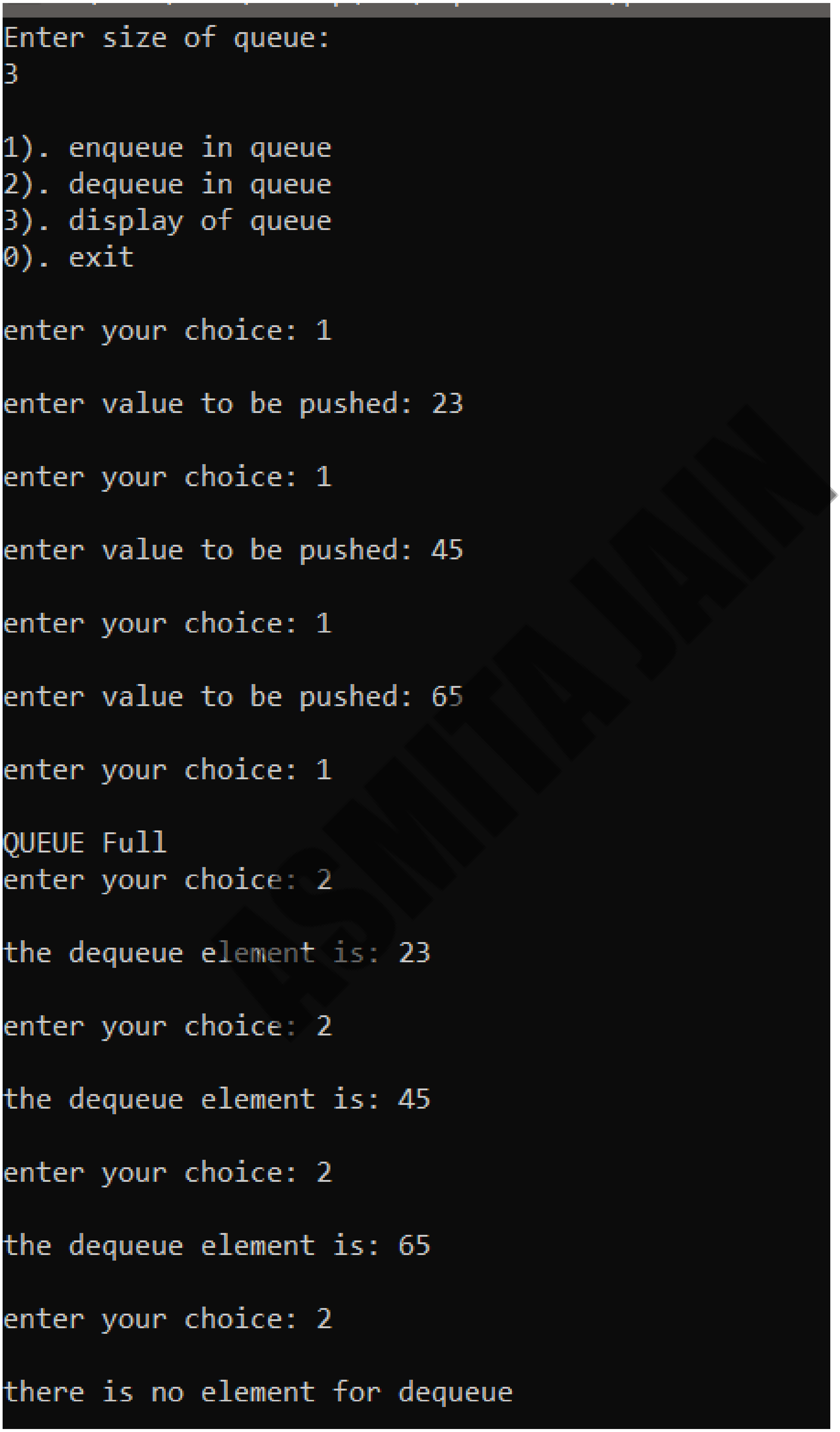
```
        {
            cout << a[i]<< endl;
        }
    }
}
int is_full()
{
    if(rear>=size-1)
        return 1;
    else
        return 0;
}
int is_empty_()
{
   if(front>rear)
        return 1;
   else
    return 0;
}
```

**(output on next page)**

## Output:



```
Enter size of queue:
5

1). enqueue in queue
2). dequeue in queue
3). display of queue
0). exit


enter your choice: 1

enter value to be pushed: 23


enter your choice: 1

enter value to be pushed: 45


enter your choice: 1

enter value to be pushed: 56


enter your choice: 3

the queue is:
23
45
56


enter your choice: 2

the dequeue element is: 23


enter your choice: 3

the queue is:
45
56
```



```
Enter size of queue:
3

1). enqueue in queue
2). dequeue in queue
3). display of queue
0). exit

enter your choice: 1

enter value to be pushed: 23

enter your choice: 1

enter value to be pushed: 45

enter your choice: 1

enter value to be pushed: 65

enter your choice: 1

QUEUE Full
enter your choice: 2

the dequeue element is: 23

enter your choice: 2

the dequeue element is: 45

enter your choice: 2

the dequeue element is: 65

enter your choice: 2

there is no element for dequeue
```

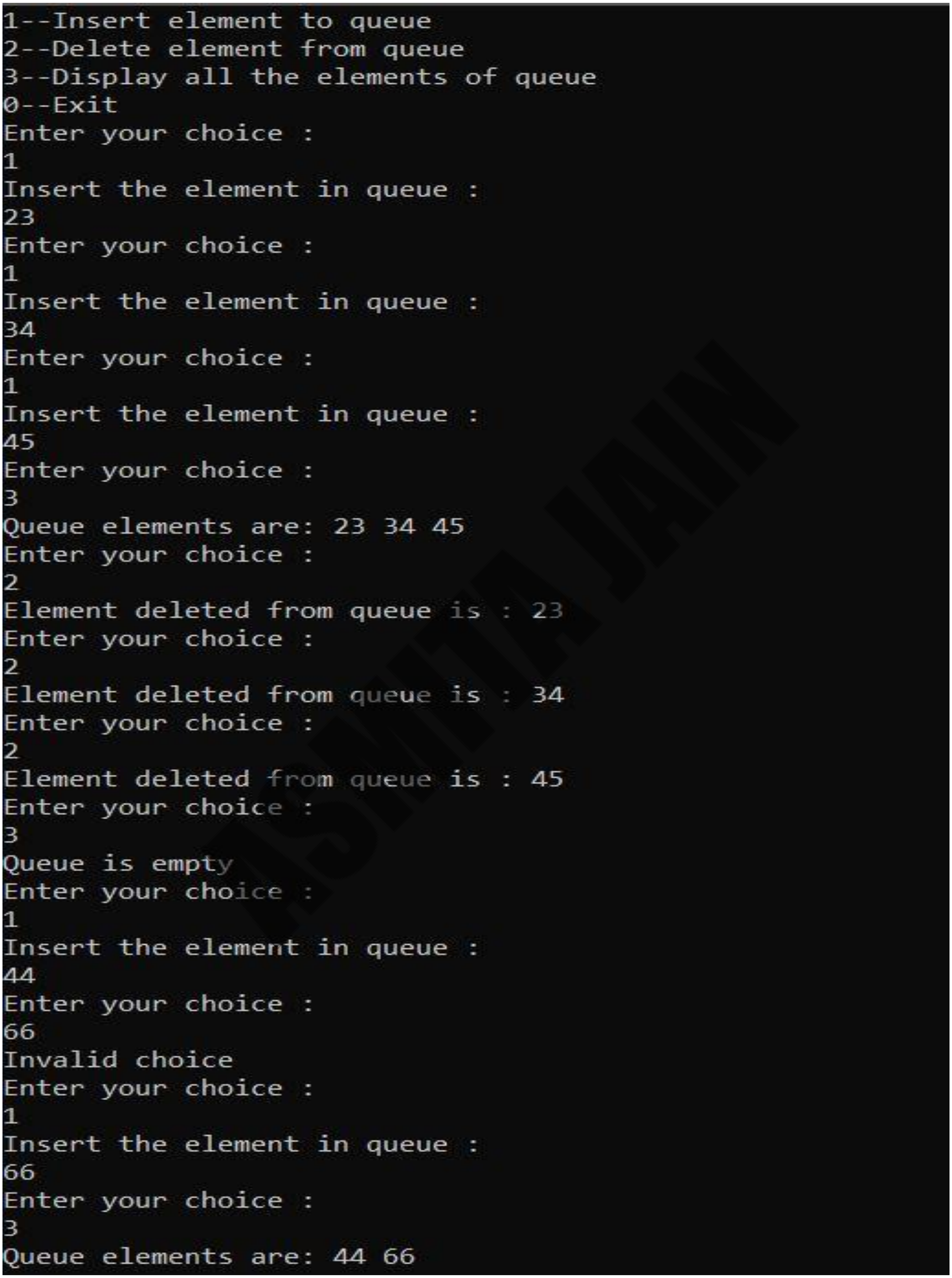
## 5.ii Queue using linked list.

**Code:**

```
#include <iostream>
#include<process.h>
using namespace std;
struct node {
  int data;
  struct node *next;
};
struct node* front = NULL;
struct node* rear = NULL;
struct node* temp;

void Insert();
void Delete();
void Display();
int main() {
  int ch;
  cout<<"1--Insert element to queue"<<endl;
  cout<<"2--Delete element from queue"<<endl;
  cout<<"3--Display all the elements of queue"<<endl;
  cout<<"0--Exit"<<endl;
  while(1)
  {
    cout<<"Enter your choice : "<<endl;
    cin>>ch;
    switch (ch) {
      case 1:
        Insert();
        break;
      case 2:
        Delete();
        break;
      case 3:
         Display();
        break;
      case 4:
         exit(0);
      default:
         cout<<"Invalid choice"<<endl;
    }
  }
  return 0;
}
void Insert() {
  int val;
  cout<<"Insert the element in queue : "<<endl;
  cin>>val;
  if (rear == NULL) {
    rear = (struct node *)malloc(sizeof(struct node));
    rear->next = NULL;
    rear->data = val;
    front = rear;
  } else {
    temp=(struct node *)malloc(sizeof(struct node));
    rear->next = temp;
    temp->data = val;
    temp->next = NULL;
    rear = temp;
  }
}
void Delete() {
  temp = front;
  if (front == NULL) {
    cout<<"Underflow"<<endl;
    return;
  }
  else
  if (temp->next != NULL) {
    temp = temp->next;
    cout<<"Element deleted from queue is : "<<front->data<<endl;
    free(front);
    front = temp;
  } else {
    cout<<"Element deleted from queue is : "<<front->data<<endl;
    free(front);
    front = NULL;
    rear = NULL;
  }
}
void Display() {
  temp = front;
  if ((front == NULL) && (rear == NULL)) {
        cout<<"Queue is empty"<<endl;
        return;
    }
    cout<<"Queue elements are: ";
    while (temp != NULL) {
        cout<<temp->data<<" ";
        temp = temp->next;
    }
    cout<<endl;
}
```

**(output on next page)**

**Output:**



```
1--Insert element to queue
2--Delete element from queue
3--Display all the elements of queue
0--Exit
Enter your choice :
1
Insert the element in queue :
23
Enter your choice :
1
Insert the element in queue :
34
Enter your choice :
1
Insert the element in queue :
45
Enter your choice :
3
Queue elements are: 23 34 45
Enter your choice :
2
Element deleted from queue is : 23
Enter your choice :
2
Element deleted from queue is : 34
Enter your choice :
2
Element deleted from queue is : 45
Enter your choice :
3
Queue is empty
Enter your choice :
1
Insert the element in queue :
44
Enter your choice :
66
Invalid choice
Enter your choice :
1
Insert the element in queue :
66
Enter your choice :
3
Queue elements are: 44 66
```

**6. Write a program that uses stack operations to convert a given infix operation to postfix operation.**

**Code:**

```
#include<iostream>
#include<string>
#define MAX 20
using namespace std;

char stk[20];
int top=-1;
void push(char oper);
char pop();
int priority ( char alpha );
string convert(string infix);



int main()
{
   int cont;
   string infix, postfix;
   cout<<"\nEnter the infix expression : ";
   cin>>infix;
   postfix = convert(infix);
   return 0;
}
void push(char oper)
```

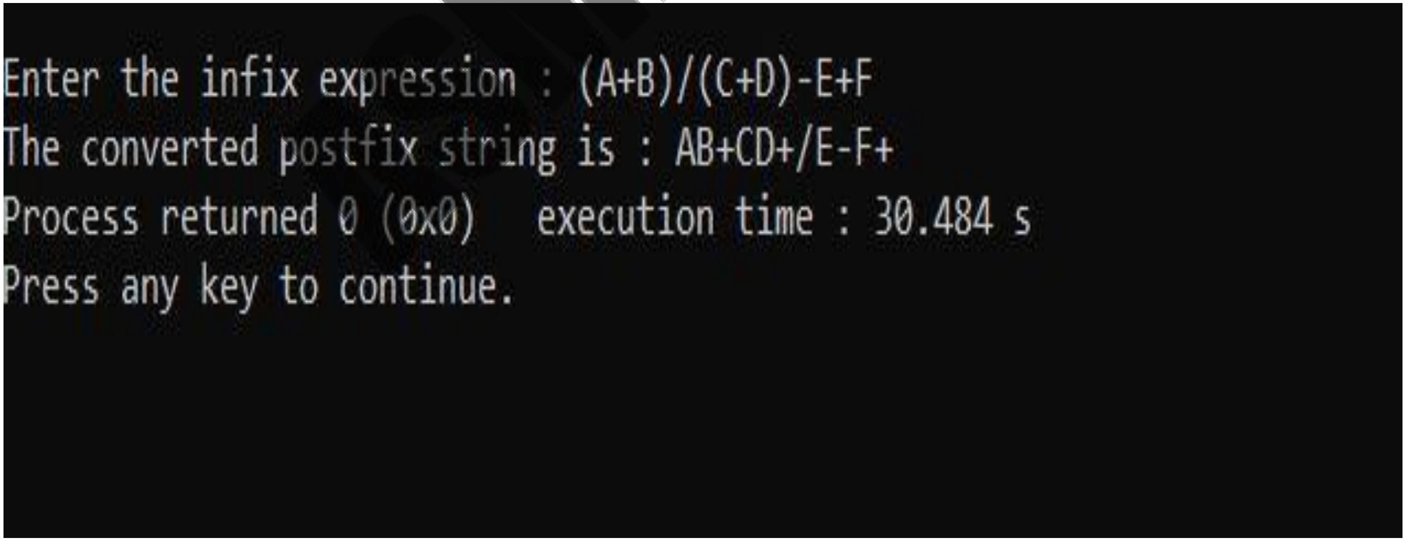
```
{
   if(top==MAX-1)
   {
      cout<<"stackfull!!!!";
   }

   else
   {
      top++;
      stk[top]=oper;
   }
}

char pop()
{
   char ch;
   if(top==-1)
   {
      cout<<"stackempty!!!!";
   }
   else
   {
      ch=stk[top];
      stk[top]='\0';
      top--;
      return(ch);
   }
   return 0;
```

```
}
int priority ( char alpha )
{
   if(alpha == '+' || alpha =='-')
   {
      return(1);
   }

   if(alpha == '*' || alpha =='/')
   {
      return(2);
   }

   if(alpha == '$')
   {
      return(3);
   }

   return 0;
}
string convert(string infix)
{
   int i=0;
   string postfix = "";
   while(infix[i]!='\0')
   {
      if(infix[i]>='a' && infix[i]<='z'|| infix[i]>='A'&& infix[i]<='Z')
      {
```

```
    postfix.insert(postfix.end(),infix[i]);
    i++;
}
else if(infix[i]=='(' || infix[i]=='{'  || infix[i]=='[')
{
    push(infix[i]);
    i++;
}
else if(infix[i]==')' || infix[i]=='}'  || infix[i]==']')
{
    if(infix[i]==')')
    {
        while(stk[top]!='(')
        {            postfix.insert(postfix.end(),pop());
        }
        pop();
        i++;
    }
    if(infix[i]==']')
    {
        while(stk[top]!='[')
        {
            postfix.insert(postfix.end(),pop());
        }
        pop();
        i++;
    }
    if(infix[i]=='}')
    {
        while(stk[top]!='{')
        {
            postfix.insert(postfix.end(),pop());
        }
        pop();
        i++;
    }
}
else
{
    if(top==-1)
    {
        push(infix[i]);
        i++;
    }

    else if( priority(infix[i]) <= priority(stk[top])) {
        postfix.insert(postfix.end(),pop());

        while(priority(stk[top]) == priority(infix[i])){
            postfix.insert(postfix.end(),pop());
            if(top < 0) {
                break;
            }
        }
        push(infix[i]);
```

```
            i++;
        }
        else if(priority(infix[i]) > priority(stk[top])) {
            push(infix[i]);
            i++;
        }
    }
  }
  while(top!=-1)
  {
    postfix.insert(postfix.end(),pop());
  }
  cout<<"The converted postfix string is : "<<postfix;
  return postfix;
}
```

**OUTPUT:**



```
Enter the infix expression : (A+B)/(C+D)-E+F
The converted postfix string is : AB+CD+/E-F+
Process returned 0 (0x0)   execution time : 30.484 s
Press any key to continue.
```

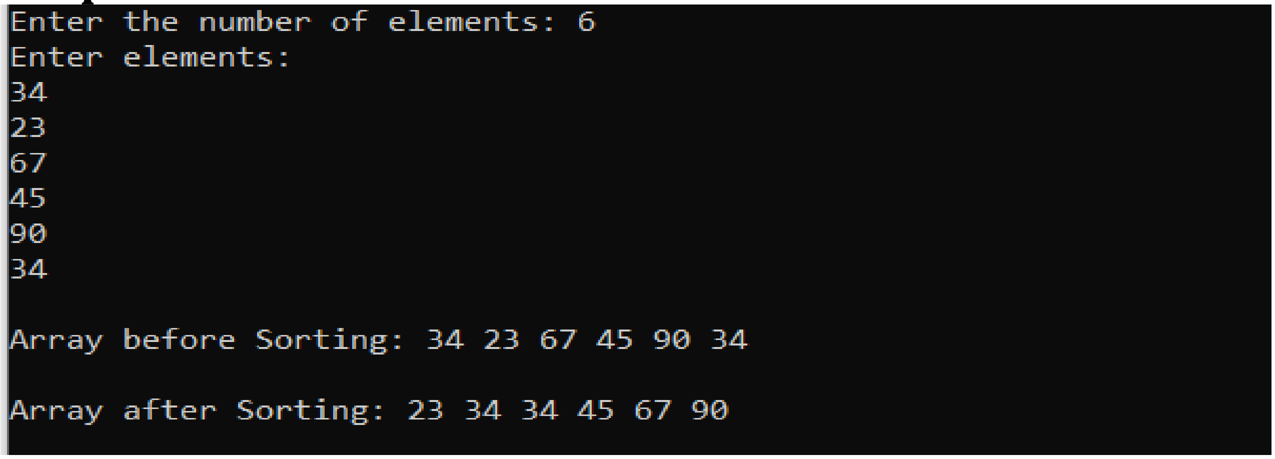
## 7.Write a program for implementing the following sorting methods to arrange a list of integer in ascending order:

### a. Bubble sort

**Code:**

```
#include<iostream>
using namespace std;
void swap_(int &a, int &b);
void display(int *array, int size);
void bubbleSort(int *array, int size);

int main()
{
  int n;
  cout << "Enter the number of elements: ";
  cin >> n;
  int arr[n];
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++)
    cin >> arr[i];

  cout << "\nArray before Sorting: ";
  display(arr, n);
  bubbleSort(arr, n);
  cout << "\nArray after Sorting: ";
  display(arr, n);
}

void swap_(int &a, int &b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void display(int *array, int size)
{
  for(int i = 0; i<size; i++)
    cout << array[i] << " ";
  cout << endl;
}
void bubbleSort(int *array, int size)
{
  for(int i = 0; i<size; i++)
  {
     int flag=0;
    for(int j = 0; j<size-i-1; j++)
    {
      if(array[j] > array[j+1])
      {
        swap_(array[j], array[j+1]);
        flag = 1;
      }
    }
    if(!flag)
      break;
  }
}
```

**Output:**



```
Enter the number of elements: 6
Enter elements:
34
23
67
45
90
34

Array before Sorting: 34 23 67 45 90 34

Array after Sorting: 23 34 34 45 67 90
```

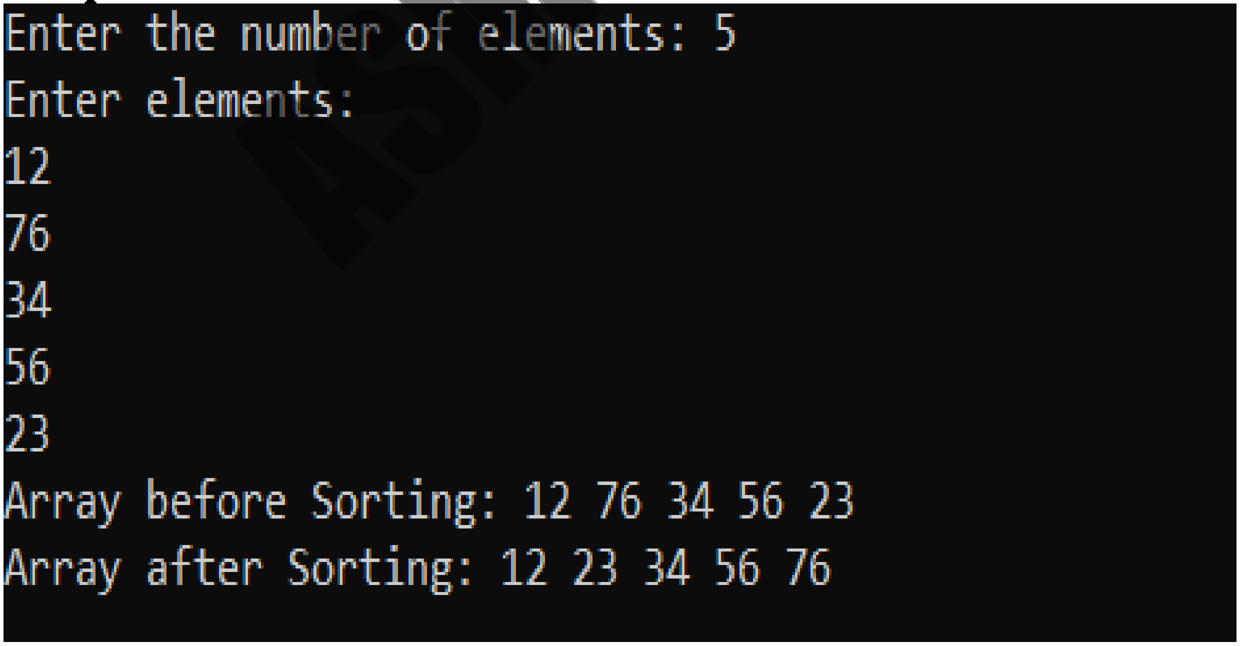
## b. Selection sort

**Code:**

```
#include<iostream>
using namespace std;
void swap_(int &a, int &b);
void display(int *array, int size);
void selectionSort(int *array, int size);

int main()
{
  int n;
  cout << "Enter the number of elements: ";
  cin >> n;
  int arr[n];
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++) {
    cin >> arr[i];
  }
  cout << "Array before Sorting: ";
  display(arr, n);
  selectionSort(arr, n);
  cout << "Array after Sorting: ";
  display(arr, n);
}
void swap_(int &a, int &b)
{
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void display(int *array, int size)
```

```
{
   for(int i = 0; i<size; i++)
      cout << array[i] << " ";
   cout << endl;
}
void selectionSort(int *array, int size)
{
   int i, j, imin;
   for(i = 0; i<size-1; i++)
    {
      imin = i;
      for(j = i+1; j<size; j++)
         if(array[j] < array[imin])
            imin = j;

         swap(array[i], array[imin]);
   }
}
```

**Output:**



```
Enter the number of elements: 5
Enter elements:
12
76
34
56
23
Array before Sorting: 12 76 34 56 23
Array after Sorting: 12 23 34 56 76
```

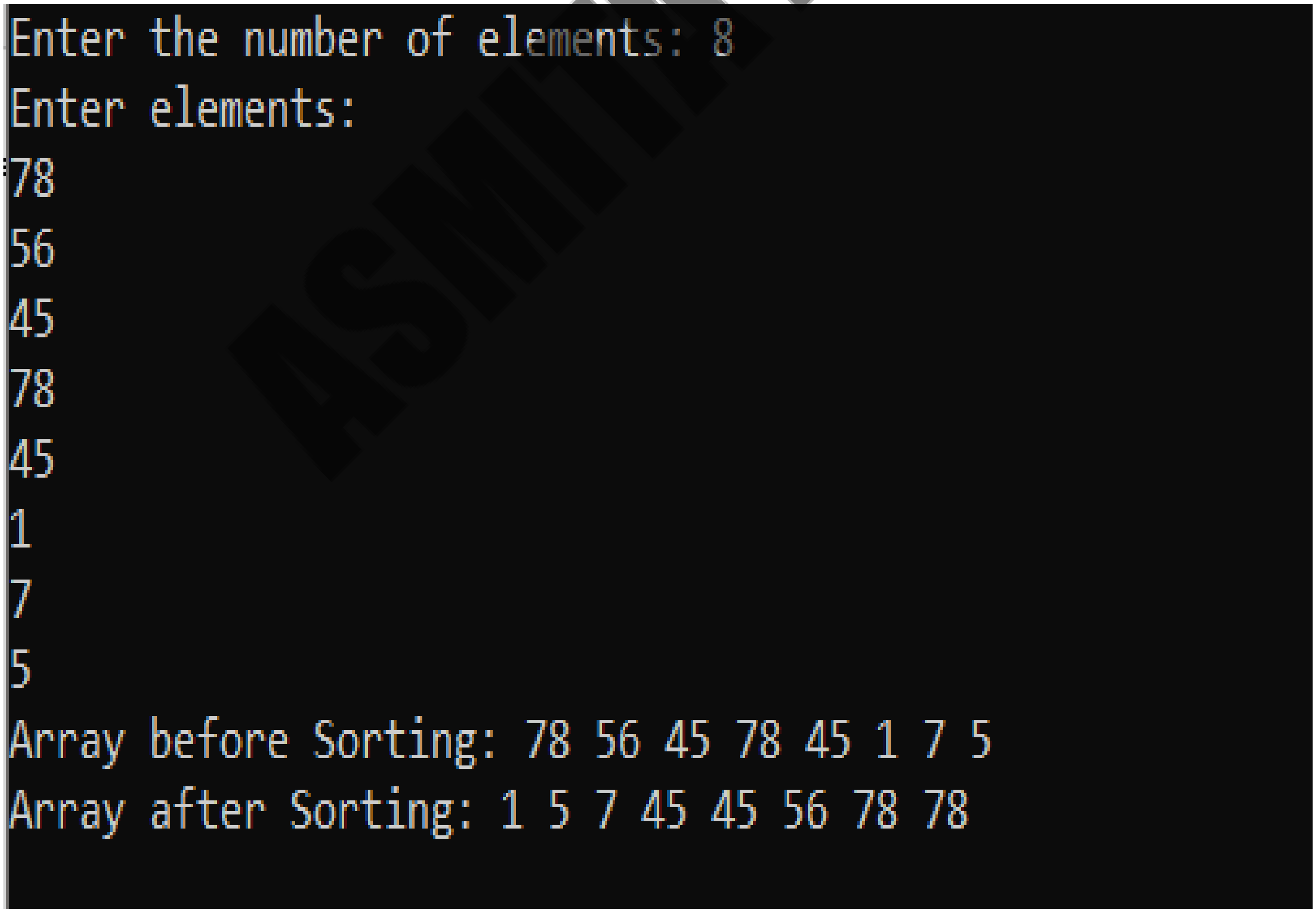
## c. Insertion sort

**Code:**

```
#include<iostream>
using namespace std;
void display(int *array, int size);
void insertionSort(int *array, int size);

int main()
{
  int n;
  cout << "Enter the number of elements: ";
  cin >> n;
  int arr[n];
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++)
    cin >> arr[i];

  cout << "Array before Sorting: ";
  display(arr, n);
  insertionSort(arr, n);
  cout << "Array after Sorting: ";
  display(arr, n);
  return 0;
}

void display(int *array, int size)
{
  for(int i = 0; i<size; i++)
    cout << array[i] << " ";
  cout << endl;
}
void insertionSort(int *array, int size)
```

```
{
  int key, j;
  for(int i = 1; i<size; i++)
   {
    key = array[i];
    j = i;
    while(j > 0 && array[j-1]>key)
    {
      array[j] = array[j-1];
      j--;
    }
    array[j] = key;
  }
}
```

**OUTPUT:**



```
Enter the number of elements: 8
Enter elements:
78
56
45
78
45
1
7
5
Array before Sorting: 78 56 45 78 45 1 7 5
Array after Sorting: 1 5 7 45 45 56 78 78
```

**8. Write a program for implementing the following sorting methods to arrange a list of integers in ascending order:**

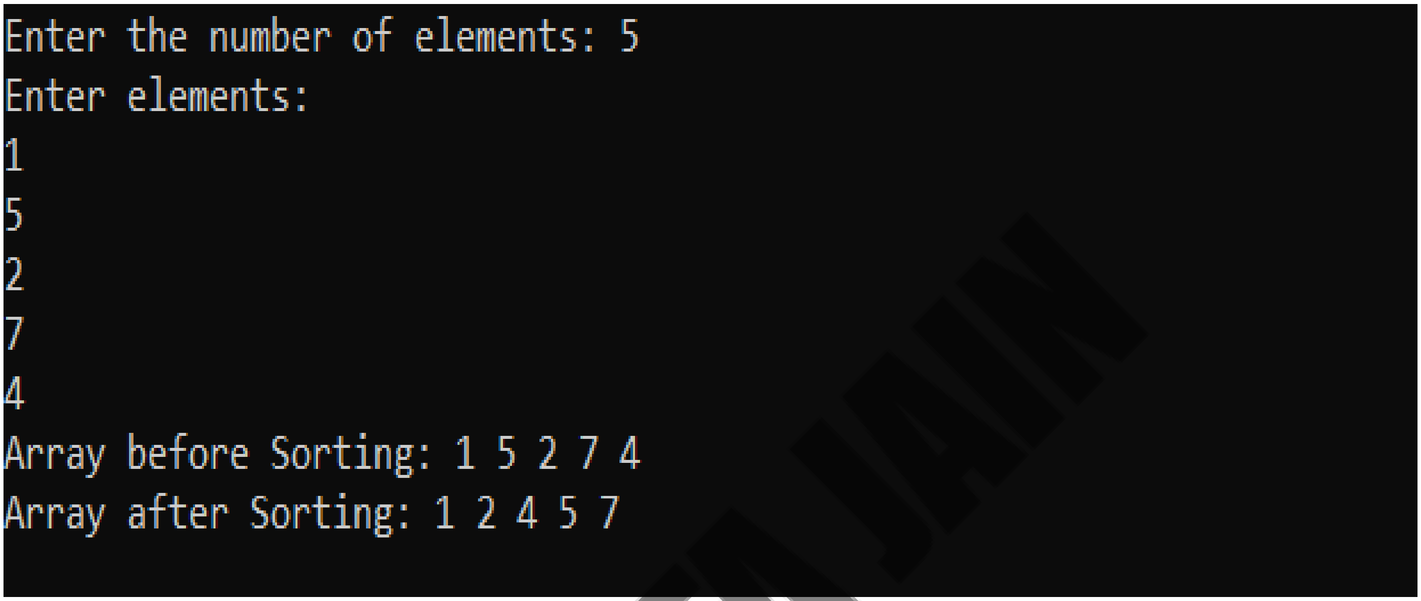
**a. Merge sort**

**Code:**

```
#include<iostream>
using namespace std;
void merge(int *array, int l, int m, int r);
void display(int *array, int size);
void swapping(int &a, int &b);
void mergeSort(int *array, int l, int r);

int main()
{
  int n;
  cout << "Enter the number of elements: ";
  cin >> n;
  int arr[n];
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++) {
    cin >> arr[i];
  }
  cout << "Array before Sorting: ";
  display(arr, n);
  mergeSort(arr, 0, n-1);
  cout << "Array after Sorting: ";
  display(arr, n);
}
```

```
void swapping(int &a, int &b)
{
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void display(int *array, int size)
{
  for(int i = 0; i<size; i++)
    cout << array[i] << " ";
  cout << endl;
}
void merge(int *array, int l, int m, int r)
{
  int i, j, k, nl, nr;
  nl = m-l+1; nr = r-m;
  int larr[nl], rarr[nr];
  for(i = 0; i<nl; i++)
    larr[i] = array[l+i];
  for(j = 0; j<nr; j++)
    rarr[j] = array[m+1+j];
  i = 0; j = 0; k = l;
  while(i < nl && j<nr)
   {
    if(larr[i] <= rarr[j])
    {
      array[k] = larr[i];
      i++;
    }
    else
    {
      array[k] = rarr[j];
      j++;
    }
    k++;
  }
  while(i<nl)
  {
    array[k] = larr[i];
    i++; k++;
  }
  while(j<nr)
   {
    array[k] = rarr[j];
    j++; k++;
  }
}
void mergeSort(int *array, int l, int r)
{
  int m;
  if(l < r)
   {
    int m = l+(r-l)/2;
    mergeSort(array, l, m);
    mergeSort(array, m+1, r);
    merge(array, l, m, r);
  }
}
```

**Output:**



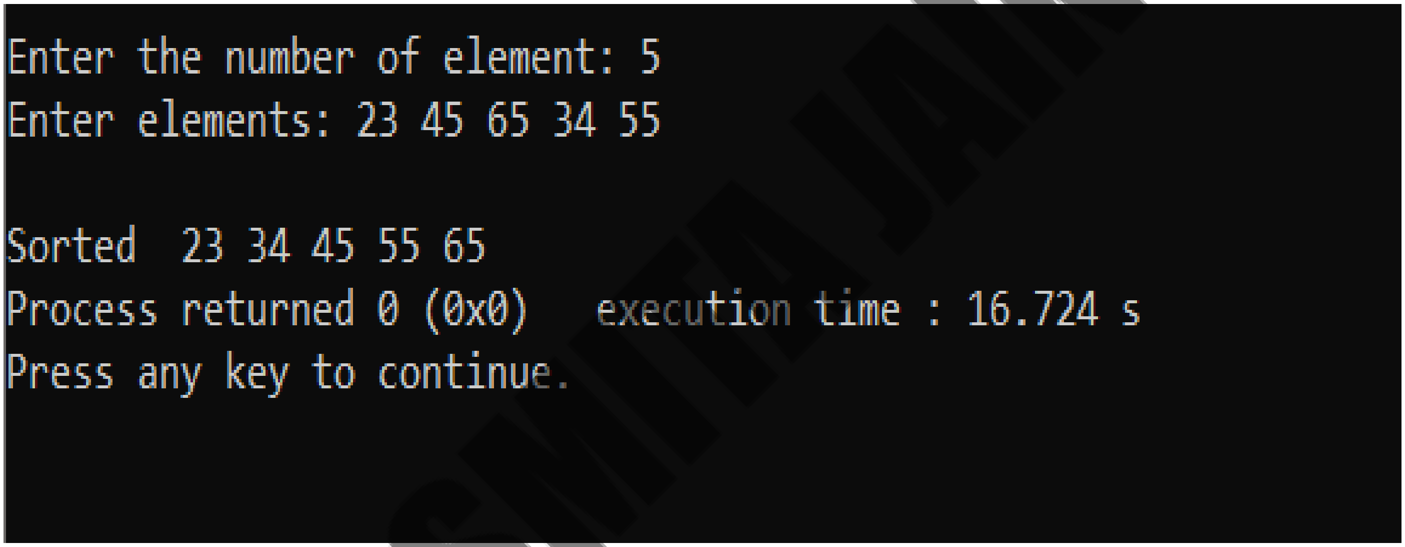
```
Enter the number of elements: 5
Enter elements:
1
5
2
7
4
Array before Sorting: 1 5 2 7 4
Array after Sorting: 1 2 4 5 7
```

**b. Quick sort**

**Code:**

```
//quick sort
#include<iostream>

using namespace std;
void swap(int *a, int *b);
int Partition(int a[], int l, int h);
int RandomPivotPartition(int a[], int l, int h);
int QuickSort(int a[], int l, int h);

int main()
{
  int n, i;
  cout<<"\nEnter the number of element: ";
  cin>>n;
  int arr[n];
  cout<<"Enter elements: ";
  for(i = 0; i < n; i++)
  {
    cin>>arr[i];
  }
  QuickSort(arr, 0, n-1);
  cout<<"\nSorted ";
  for (i = 0; i < n; i++)
    cout<<" "<<arr[i];
  return 0;
}

void swap(int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

int Partition(int a[], int l, int h)
{
  int pivot, index, i;
  index = l;
  pivot = h;
  for(i = l; i < h; i++)
   {
    if(a[i] < a[pivot])
    {
      swap(&a[i], &a[index]);
      index++;
    }
  }
  swap(&a[pivot], &a[index]);
  return index;
}
int RandomPivotPartition(int a[], int l, int h)
{
  int pvt, n, temp;
  n = rand();
  pvt = l + n%(h-l+1);
  swap(&a[h], &a[pvt]);
  return Partition(a, l, h);
}
int QuickSort(int a[], int l, int h)
{
  int pindex;
  if(l < h)
   {
    pindex = RandomPivotPartition(a, l, h);
    QuickSort(a, l, pindex-1);
    QuickSort(a, pindex+1, h);
  }
  return 0;
}
```

**OUTPUT:**



```
Enter the number of element: 5
Enter elements: 23 45 65 34 55

Sorted  23 34 45 55 65
Process returned 0 (0x0)   execution time : 16.724 s
Press any key to continue.
```

**9. Write a program to perform the following using functions:**

**a. Create a binary search tree of characters.**

**b. Traverse the above binary search tree recursively in postorder**

**c. Count the number of nodes in the above tree**

**CODE:**

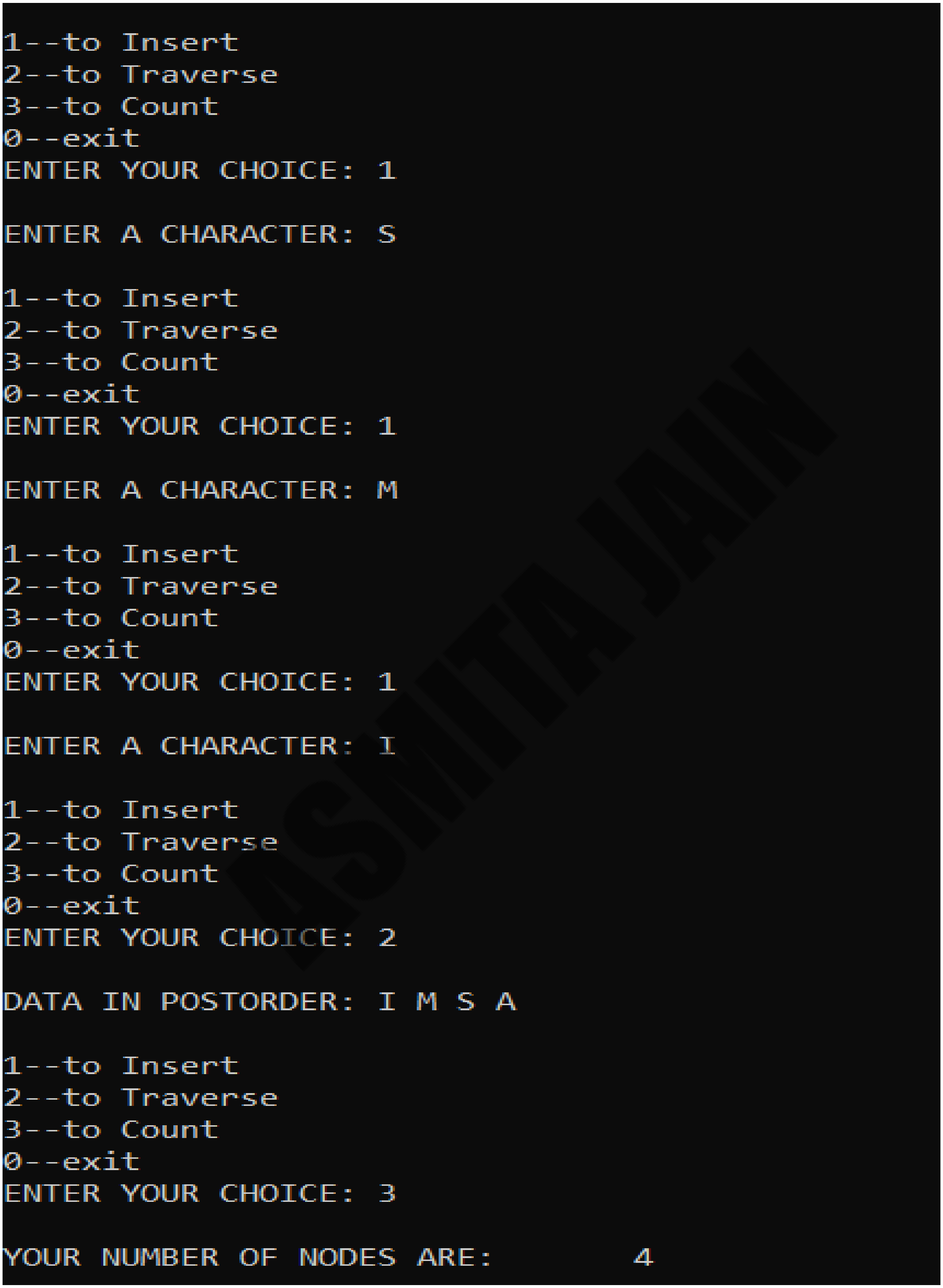
```
#include<iostream>
using namespace std;
class Binary_tree
{
        private:
                Binary_tree *left;
                char Data;
                Binary_tree *right;
        public:
        static Binary_tree* head;
        int cnt=0;
        bool insert(char d)
        {Binary_tree *New=new Binary_tree;
         Binary_tree *next=new Binary_tree;
           if(head==NULL)
                {
                  New->left=head;
                  New->Data=d;
                  cnt++;
                  New->right=head;
                  head=New;
                }
```

```
 else
 {
     New=head;
   next->Data=d;next->left=NULL;next->right=NULL;cnt++;
          while(New->left!=NULL||New->right!=NULL)
          {
               if(d<New->Data)
          {
           if(New->left==NULL)
                {
                 New->left=next;
              return 0;
                }
                New=New->left;
            }
               else
               {
                if(New->right==NULL)
                {
                 New->right=next;
              return 0;
                }
                New=New->right;
            }
}
if(d<New->Data){New->left=next;}
     else New->right=next;
 }
```

```
                return 0;
        }
        void traverse(Binary_tree *bt=head)
        {
                if(head==NULL)
                {
                        cout<<"\nEMPTY! ENTER A NODE FIRST! - \n";
                }
                else
                {
                if(bt==NULL)
                return;
                traverse(bt->left);
                traverse(bt->right);
                cout<<bt->Data<<" ";
           }
   }
};
Binary_tree* Binary_tree::head=NULL;
int main()
{
        Binary_tree tree;
        char data;
   short int  choice;
   while(1)
   {
       cout<<" \n1--to Insert\n2--to Traverse \n3--to Count \n0--
exit\n";
```

```
      cout<<"ENTER YOUR CHOICE: ";
      cin>>choice;
      switch(choice)
      {
      case 1:
         cout<<"\nENTER A CHARACTER: ";
         cin>>data;
         tree.insert(data);
         break;
      case 2:
         cout<<"\nDATA IN POSTORDER: ";
         tree.traverse();
         cout<<endl;
         break;
      case 3:
         cout<<"\nYOUR NUMBER OF NODES ARE:\t"<<tree.cnt<<endl;
         break;
      case 4:
         exit(0);
      default:
         cout<<"\nWRONG CHOICE \n";
      }

   }
       return 0;
}
```

# OUTPUT:



```
1--to Insert
2--to Traverse
3--to Count
0--exit
ENTER YOUR CHOICE: 1

ENTER A CHARACTER: S

1--to Insert
2--to Traverse
3--to Count
0--exit
ENTER YOUR CHOICE: 1

ENTER A CHARACTER: M

1--to Insert
2--to Traverse
3--to Count
0--exit
ENTER YOUR CHOICE: 1

ENTER A CHARACTER: I

1--to Insert
2--to Traverse
3--to Count
0--exit
ENTER YOUR CHOICE: 2

DATA IN POSTORDER: I M S A

1--to Insert
2--to Traverse
3--to Count
0--exit
ENTER YOUR CHOICE: 3

YOUR NUMBER OF NODES ARE:       4
```

**10. Write a program to perform the following using functions:**
**a. Create a binary search tree of integers.**
**b. Traverse the above binary search tree recursively in inorder.**
**c. Count the number of nodes in the above tree.**

**CODE:**

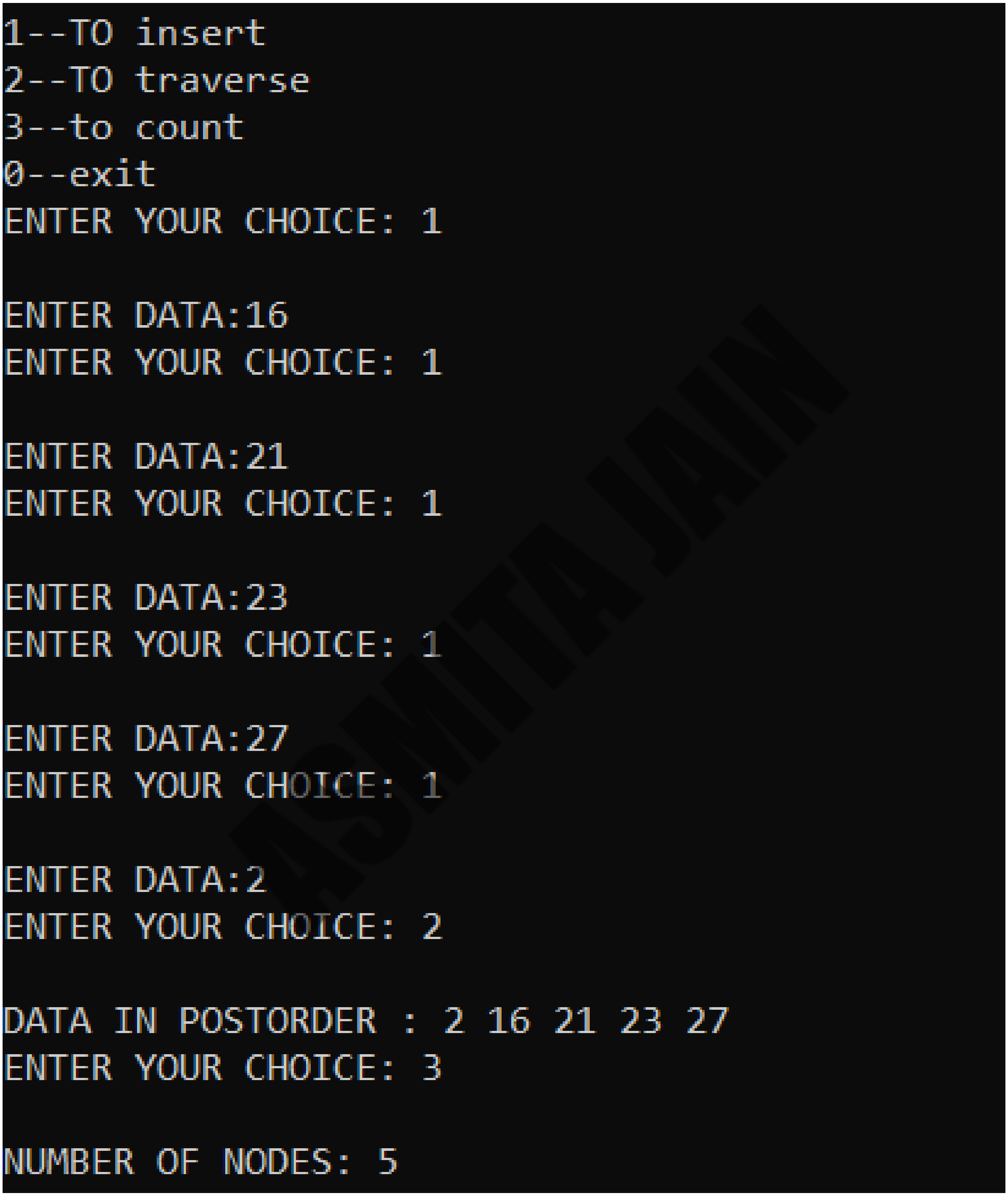
```
#include<iostream>
#include<process.h>
using namespace std;
class Binary_tree
{
        private:
                Binary_tree *left;
                int Data;
                Binary_tree *right;
        public:
        static Binary_tree* head;
        int cnt=0;
        bool insert(int d)
        {Binary_tree *New=new Binary_tree;
         Binary_tree *next=new Binary_tree;
           if(head==NULL)
                {
                  New->left=head;
                  New->Data=d;
                  cnt++;
                  New->right=head;
```

```
   head=New;
 }
 else
 {
      New=head;
   next->Data=d;next->left=NULL;next->right=NULL;cnt++;
           while(New->left!=NULL||New->right!=NULL)
           {
                if(d<New->Data)
           {
            if(New->left==NULL)
                 {
                  New->left=next;
               return 0;
                 }
                 New=New->left;
             }
                else
                {
                 if(New->right==NULL)
                 {
                  New->right=next;
               return 0;
                 }
                 New=New->right;
             }
}
if(d<New->Data){New->left=next;}
```

```
                    else New->right=next;
                }
                return 0;
        }
        void traverse(Binary_tree *bt=head)
        {
                if(head==NULL)
                {
                        cout<<"\nEMPTY\n";
                }
                else
                {
                if(bt==NULL)
                return;
                traverse(bt->left);
                cout<<bt->Data<<" ";
                traverse(bt->right);
                }
    }
};

Binary_tree* Binary_tree::head=NULL;

int main()
{
        Binary_tree tree;
        int data;
    short int  choice;
```

```
  cout<<" \n1--TO insert\n2--TO traverse \n3--to count \n0--exit\n";
  while(1)
  {
    cout<<"ENTER YOUR CHOICE: ";
    cin>>choice;
    switch(choice)
    {
    case 1:
      cout<<"\nENTER DATA:";
      cin>>data;
      tree.insert(data);
      break;
    case 2:
      cout<<"\nDATA IN POSTORDER : ";
      tree.traverse();
      cout<<endl;
      break;
    case 3:
      cout<<"\nNUMBER OF NODES: "<<tree.cnt<<endl;
      break;
    case 0:
      exit(1);
    default:
      cout<<"\nWRONG CHOICE\n";
     }
  }
  return 0;
}
```

**OUTPUT:**



```
1--TO insert
2--TO traverse
3--to count
0--exit
ENTER YOUR CHOICE: 1

ENTER DATA:16
ENTER YOUR CHOICE: 1

ENTER DATA:21
ENTER YOUR CHOICE: 1

ENTER DATA:23
ENTER YOUR CHOICE: 1

ENTER DATA:27
ENTER YOUR CHOICE: 1

ENTER DATA:2
ENTER YOUR CHOICE: 2

DATA IN POSTORDER : 2 16 21 23 27
ENTER YOUR CHOICE: 3

NUMBER OF NODES: 5
```