MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

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

DATA STRUCTURE LAB

ASSIGNMENT 3

SUBMITTED BY:

ASMITA JAIN

0901EO201017

SUBMITTED TO:

PROF. NAMRATA AGRAWAL

Introduction:

1. 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()
{
```

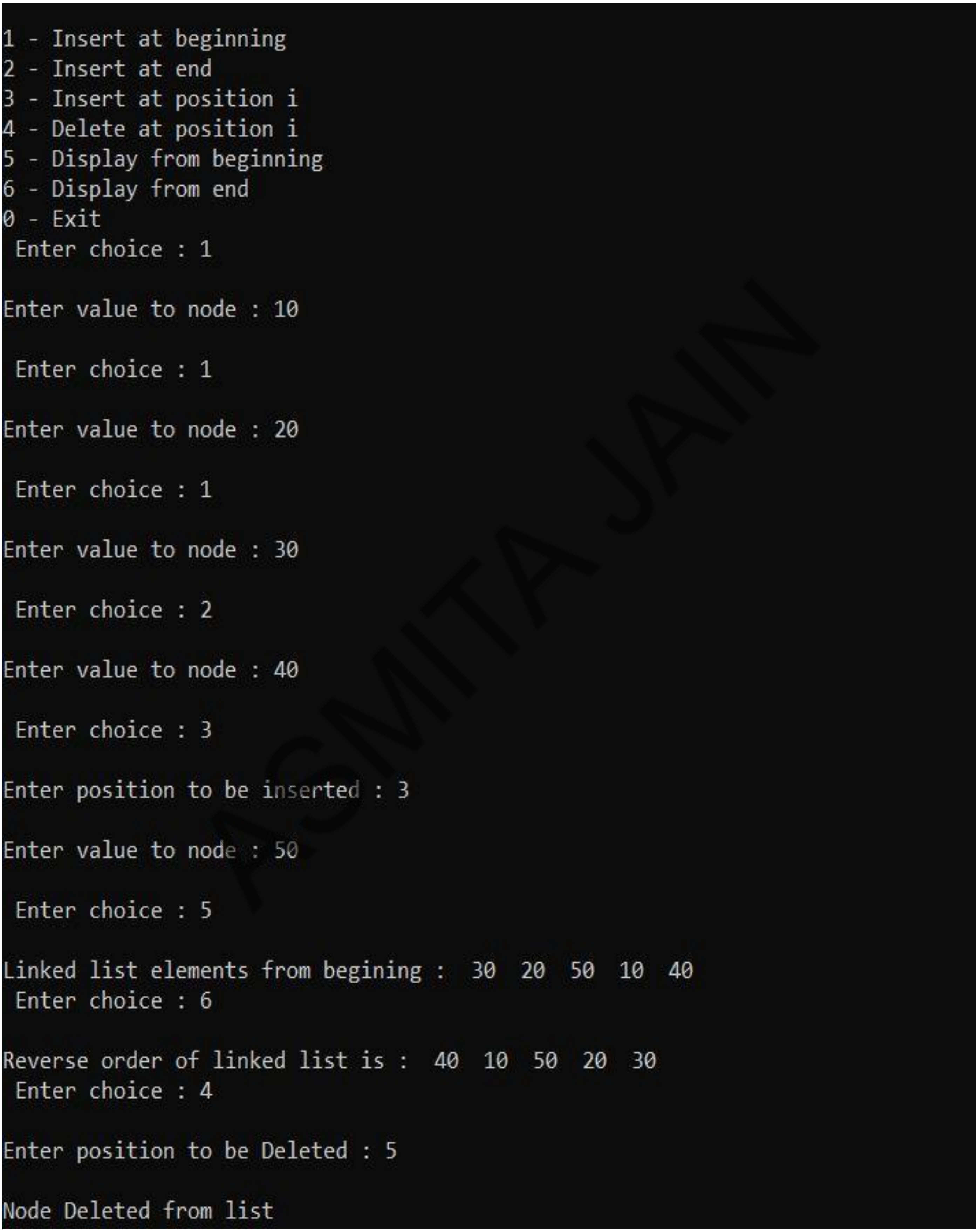
```
int pos, i = 2;

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 ON NEXT PAGE)

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
```

2. Write a program to implement stack using array.

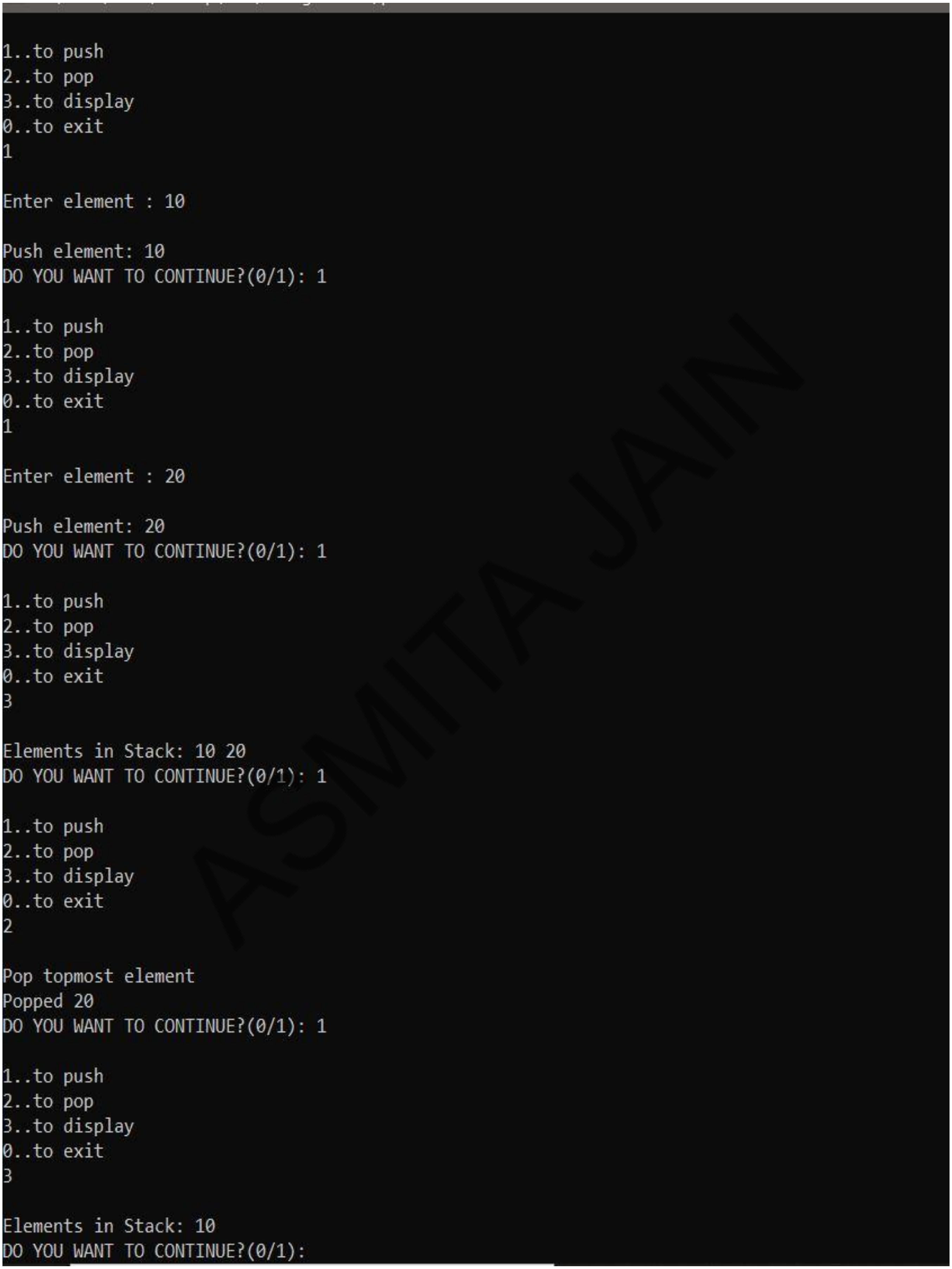
Code:

```
#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):
```

3. Write a program to implement stack using linked list.

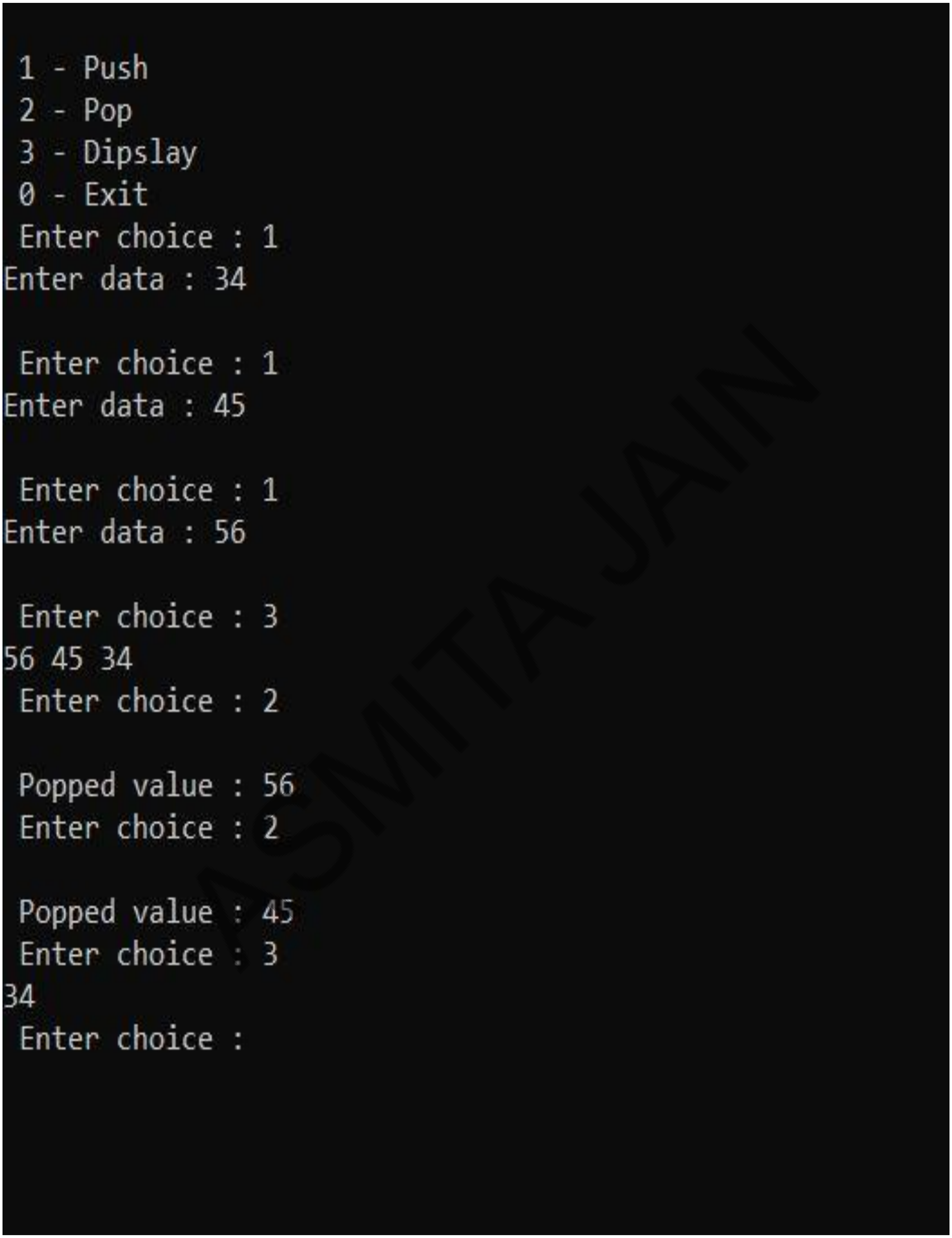
Code:

```
#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 :
```