

**PRACTICE PROBLEM-2**

NAME : NITHISH.G

REG NO : 19BCS0012

SUBJECT : DATA STRUCTURE LAB

COURSE : CSC2001

(CODE)

SLOT : L37+L38

Prof. : MARY MEKALA A

1. Create a menu driven program for to implement a singly linked list dynamically and perform the following operations:
2. Create a list with 4 nodes.
3. Insert an element at the beginning.
4. Insert an element in between any two nodes.
5. Insert an element at the end.
6. Delete the first node.
7. Delete an element in between any two nodes.
8. Delete the last node.
9. Display the linked list.
10. Exit.

**Source code:**

**#include<stdlib.h>**

**#include <stdio.h>**

**#include<conio.h>**

**void Create\_node();**

**void Insert\_an\_element\_at\_the\_beginning();**

**void Insert\_at\_element\_between\_two\_nodes();**

**void Insert\_at\_the\_end();**

**void Delete\_the\_frist\_node();**

**void Delete\_an\_element\_in\_between\_any\_two\_nodes();**

**void Delete\_the\_last\_node();**

**void Display\_the\_linkes\_list();**

**struct node**

**{**

**int info;**

**struct node \*next;**

**};**

**struct node \*start=NULL;**

**int main()**

**{**

**int choice;**

**printf("\n------------------------------------[ REG NO : 19BCS0012 ]-------------------\n");**

**printf("\n------------------------------------[ NAME : NITHISH.G ]-------------------\n");**

**while(1){**

**printf("\n-----------------------------------------------------------------------------\n");**

**printf("\n MENU \n");**

**printf("\n 1.Create node \n");**

**printf("\n 2.Insert an element at the beginning\n");**

**printf("\n 3.Insert at element between two nodes \n");**

**printf("\n 4.Insert at the end \n");**

**printf("\n 5.Delete the frist node \n");**

**printf("\n 6.Delete an element in between any two nodes \n");**

**printf("\n 7.Delete the last node \n");**

**printf("\n 8.Display the linkes list \n");**

**printf("\n 9.Exit \n");**

**printf("\n----------------------------------------------------------------------------\n");**

**printf("Enter your choice:\t");**

**scanf("%d",&choice);**

**switch(choice)**

**{**

**case 1:**

**Create\_node();**

**break;**

**case 2:**

**Insert\_an\_element\_at\_the\_beginning();**

**break;**

**case 3:**

**Insert\_at\_element\_between\_two\_nodes();**

**break;**

**case 4:**

**Insert\_at\_the\_end();**

**break;**

**case 5:**

**Delete\_the\_frist\_node();**

**break;**

**case 6:**

**Delete\_an\_element\_in\_between\_any\_two\_nodes();**

**break;**

**case 7:**

**Display\_the\_linkes\_list();**

**break;**

**case 8:**

**Display\_the\_linkes\_list();**

**break;**

**case 9:**

**exit(0);**

**break;**

**default:**

**printf("\n Wrong Choice:\n");**

**break;**

**}**

**}**

**getch();**

**return 0;**

**}**

**void Create\_node()**

**{**

**int i,n;**

**printf(" \n\n enter the size of nodes : ");**

**scanf("%d",&n);**

**for(i=0;i<n;i++)**

**{**

**struct node \*temp,\*ptr;**

**temp=(struct node \*)malloc(sizeof(struct node));**

**if(temp==NULL)**

**{**

**printf("\n Out of Memory Space: \n");**

**exit(0);**

**}**

**printf("\n\n Enter the %d node data value : \t",i+1);**

**scanf("%d",&temp->info);**

**temp->next=NULL;**

**if(start==NULL)**

**{**

**start=temp;**

**}**

**else**

**{**

**ptr=start;**

**while(ptr->next!=NULL)**

**{**

**ptr=ptr->next;**

**}**

**ptr->next=temp;**

**}**

**}**

**printf("\n\n All %d node data values loaded and linked successfully \n\n",n);**

**printf("\n----------------------------------------------------------------------------\n");**

**Display\_the\_linkes\_list();**

**}**

**void Display\_the\_linkes\_list()**

**{**

**int i=1;**

**struct node \*ptr;**

**if(start==NULL)**

**{**

**printf("\nList is empty:\n");**

**return;**

**}**

**else**

**{**

**ptr=start;**

**printf("\nThe List elements are:\n\n");**

**while(ptr!=NULL)**

**{**

**printf("%d). %d\t",i,ptr->info );**

**ptr=ptr->next ;**

**i++;**

**}**

**}**

**}**

**void Insert\_an\_element\_at\_the\_beginning()**

**{**

**struct node \*temp;**

**temp=(struct node \*)malloc(sizeof(struct node));**

**if(temp==NULL)**

**{**

**printf("\nOut of Memory Space:\n");**

**return;**

**}**

**printf("\nEnter the data value for the node:\t" );**

**scanf("%d",&temp->info);**

**temp->next =NULL;**

**if(start==NULL)**

**{**

**start=temp;**

**}**

**else**

**{**

**temp->next=start;**

**start=temp;**

**}**

**Display\_the\_linkes\_list();**

**}**

**void Insert\_at\_the\_end()**

**{**

**struct node \*temp,\*ptr;**

**temp=(struct node \*)malloc(sizeof(struct node));**

**if(temp==NULL)**

**{**

**printf("\nOut of Memory Space:\n");**

**return;**

**}**

**printf("\nEnter the data value for the node:\t" );**

**scanf("%d",&temp->info );**

**temp->next =NULL;**

**if(start==NULL)**

**{**

**start=temp;**

**}**

**else**

**{**

**ptr=start;**

**while(ptr->next !=NULL)**

**{**

**ptr=ptr->next ;**

**}**

**ptr->next =temp;**

**}**

**Display\_the\_linkes\_list();**

**}**

**void Insert\_at\_element\_between\_two\_nodes()**

**{**

**struct node \*ptr,\*temp;**

**int i,pos;**

**temp=(struct node \*)malloc(sizeof(struct node));**

**if(temp==NULL)**

**{**

**printf("nOut of Memory Space:n");**

**return;**

**}**

**printf("\nEnter the position for the new node to be inserted:\t");**

**scanf("%d",&pos);**

**printf("\nEnter the data value of the node:\t");**

**scanf("%d",&temp->info) ;**

**temp->next=NULL;**

**if(pos==1)**

**{**

**temp->next=start;**

**start=temp;**

**}**

**else**

**{**

**for(i=0,ptr=start;i<pos-1;i++) { ptr=ptr->next;**

**if(ptr==NULL)**

**{**

**printf("\nPosition not found: \n");**

**return;**

**}**

**}**

**temp->next =ptr->next ;**

**ptr->next=temp;**

**}**

**Display\_the\_linkes\_list();**

**}**

**void Delete\_the\_frist\_node()**

**{**

**struct node \*ptr;**

**if(ptr==NULL)**

**{**

**printf("\nList is Empty:\n");**

**return;**

**}**

**else**

**{**

**ptr=start;**

**start=start->next ;**

**printf("\nThe deleted element is :%d\t\n",ptr->info);**

**free(ptr);**

**}**

**Display\_the\_linkes\_list();**

**}**

**void Delete\_the\_last\_node()**

**{**

**struct node \*temp,\*ptr;**

**if(start==NULL)**

**{**

**printf("\nList is Empty:\n");**

**exit(0);**

**}**

**else if(start->next ==NULL)**

**{**

**ptr=start;**

**start=NULL;**

**printf("\nThe deleted element is:%d\t\n",ptr->info);**

**free(ptr);**

**}**

**else**

**{**

**ptr=start;**

**while(ptr->next!=NULL)**

**{**

**temp=ptr;**

**ptr=ptr->next;**

**}**

**temp->next=NULL;**

**printf("\nThe deleted element is:%d\t\n",ptr->info);**

**free(ptr);**

**}**

**Display\_the\_linkes\_list();**

**}**

**void Delete\_an\_element\_in\_between\_any\_two\_nodes()**

**{**

**int i,pos;**

**struct node \*temp,\*ptr;**

**if(start==NULL)**

**{**

**printf("\nThe List is Empty:\n");**

**exit(0);**

**}**

**else**

**{**

**printf("\nEnter the position of the node to be deleted:\t");**

**scanf("%d",&pos);**

**if(pos==1)**

**{**

**ptr=start;**

**start=start->next ;**

**printf("\n\nThe deleted element is:%d\t\n",ptr->info );**

**free(ptr);**

**}**

**else**

**{**

**ptr=start;**

**for(i=0;i<pos;i++) { temp=ptr; ptr=ptr->next ;**

**if(ptr==NULL)**

**{**

**printf("\n\nPosition not Found:\n");**

**return; } }**

**temp->next =ptr->next ;**

**printf("\nThe deleted element is:%d\t\n\n",ptr->info );**

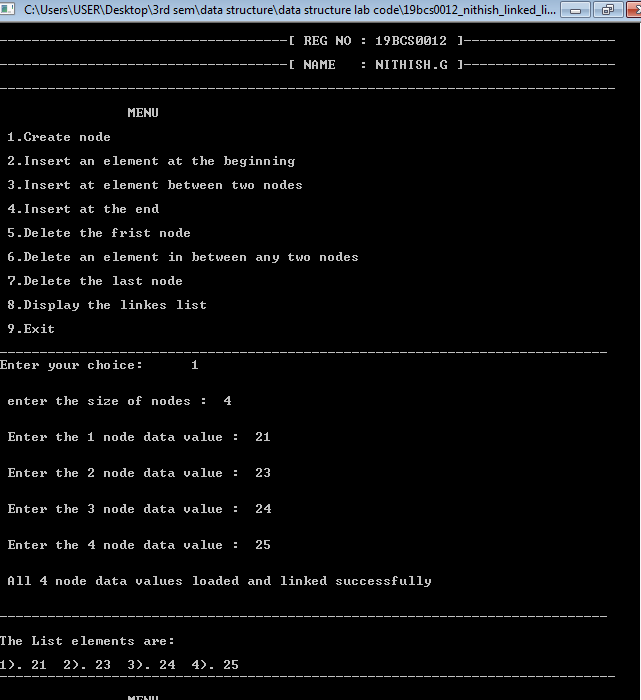
**free(ptr); } }**

**Display\_the\_linkes\_list();**

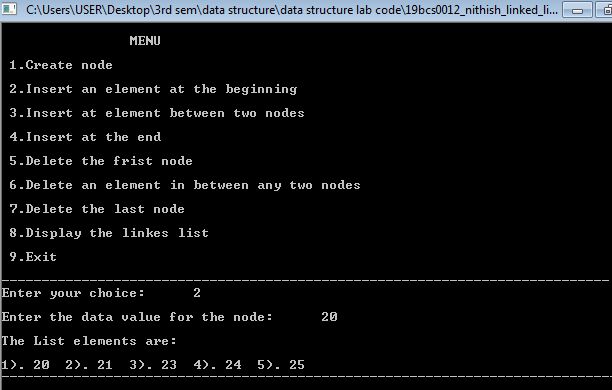
**}**

**Output**

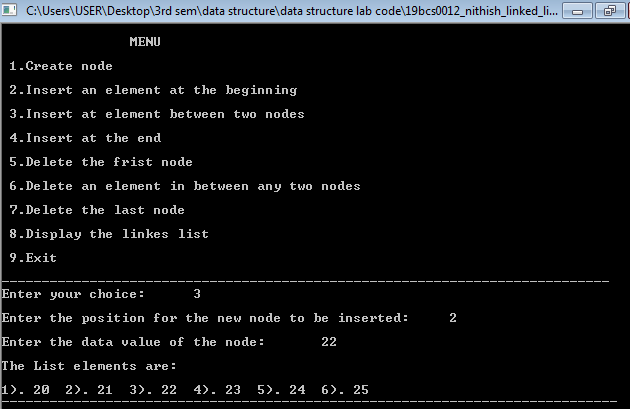
1. **Create a list with 4 nodes.**



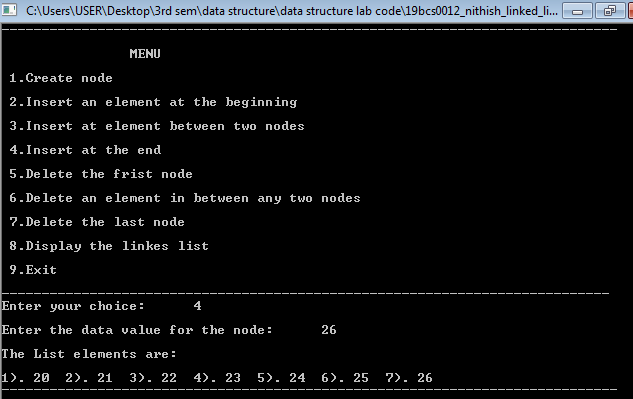
1. **Insert an element at the beginning.**



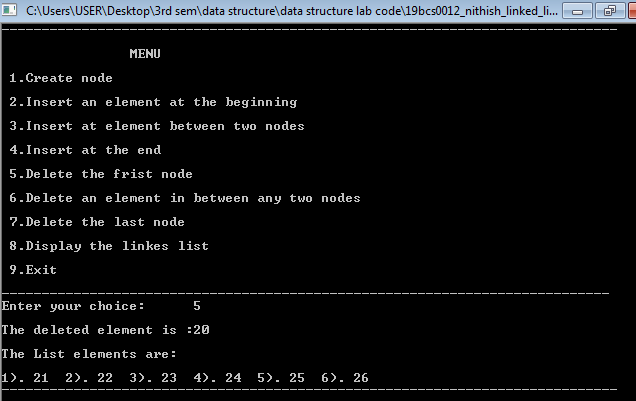
1. **Insert an element in between any two nodes.**



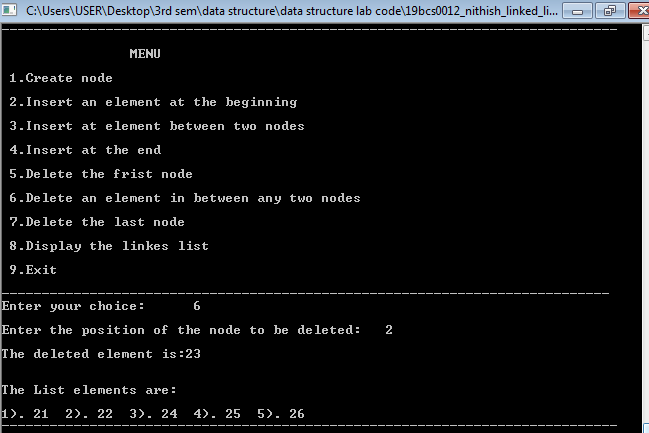
1. **Insert an element at the end.**



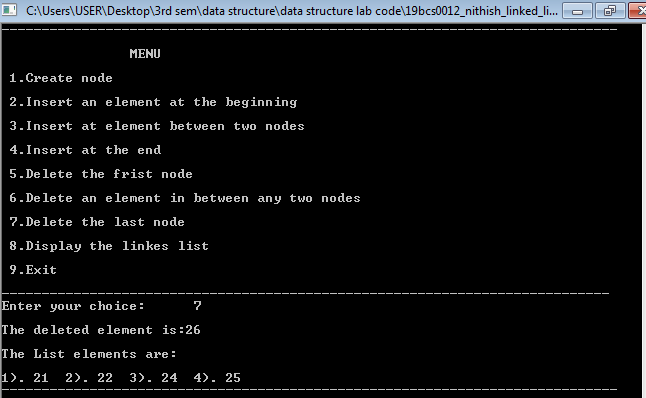
1. **Delete the first node.**



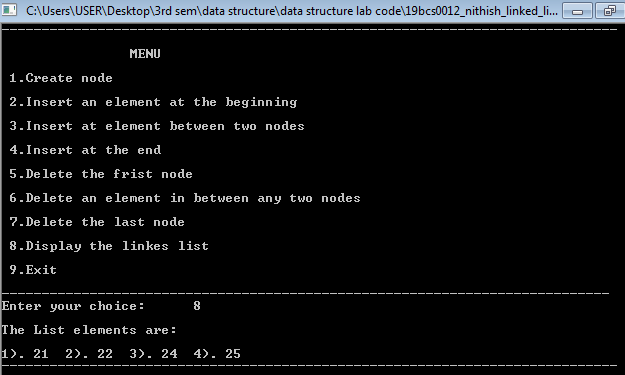
1. **Delete an element in between any two nodes.**



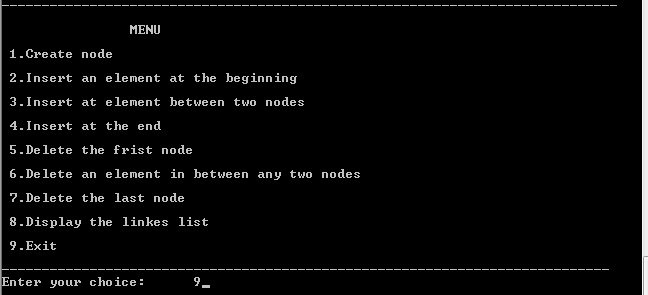
1. **Delete the last node.**



1. **Display the linked list.**



1. **Exit.**



1. **Create a menu driven program to implement stack data structure using array and perform the following operations:**

A. create a stack

B. inserts an element

C. deleting element

D. display all elements

E. display the topmost element

f. Exit.

**Source code**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 100

int stack[SIZE];

int top = -1;

void intsert\_an\_element(int element);

int deleting\_element();

void display\_top\_most\_element();

void dispaly\_all\_elements();

int main()

{

int choice, data,i=0,n;

printf("-----------------------------------------[REG NO : 19BCS0012]---\n");

printf("-----------------------------------------[NAME : NITHISH.G]---\n");

printf("[QUESTION NO - 2 ]----------------------------------------------\n");

while(1)

{

printf("----------------------------------------------------------------\n\n");

printf("1. creat\_a\_stack(push)\n\n");

printf("2. intsert\_an\_element(push)\n\n");

printf("3. deleting\_element(pop)\n\n");

printf("4. dispaly\_all\_elements\n\n");

printf("5. display\_top\_most\_element(Peek)\n\n");

printf("6. Size\n\n");

printf("7. Exit\n\n");

printf("----------------------------------------------------------------\n\n");

printf("Enter your choice: ");

scanf("\t\t%d", &choice);

switch(choice)

{

case 1:

printf("\n\nenter the size of elements to push into stack\n\n ");

scanf("%d",&n);

for(i=0;i<n;i++){

printf("\nEnter data to push into stack: ");

scanf("%d", &data);

create\_a\_stack(data);

}printf("\n\nall %d Data pushed to stack.\n",n);

dispaly\_all\_elements();

break;

case 2:

printf("\nEnter data to push into stack: ");

scanf("%d", &data);

intsert\_an\_element(data);

break;

case 3:

data = deleting\_element();

printf("\nDELETED DATA => %d\n", data);

dispaly\_all\_elements();

break;

case 4:

dispaly\_all\_elements();

break;

case 5:

display\_top\_most\_element();

break;

case 6:

printf("\nStack size: %d\n", top + 1);

break;

case 7:

printf("\nExiting.\n");

exit(0);

break;

default:

printf("\n\nInvalid choice, please try again.\n");

}

printf("\n\n");

}

return 0;

}

void dispaly\_all\_elements()

{

int i;

if(top<0)

{

printf("\n\nTHE STACK IS EMPTY\n\n");

}

else

{

printf("\nTHE ELEMENTS FROM TOP TO BOTTOM: ");

for( i = top; i>=0; i--)

{

printf("%d ",stack[i]);

}

}

}

int deleting\_element()

{

if (top < 0)

{

printf("\nStack is empty.\n");

return 0;

}

return stack[top--];

}

void display\_top\_most\_element()

{

if(top<0)

{

printf("\nSTACK IS EMPTY\n");

}

else

{

printf("\nTHE TOP ELEMENT IS %d\n",stack[top]);

}

}

void intsert\_an\_element(int element)

{

if (top >= SIZE)

{

printf("\nStack Overflow, can't add more element element to stack.\n");

}

top++;

stack[top] = element;

}

int create\_a\_stack(int element)

{

if (top >= SIZE)

{

printf("\nStack Overflow, can't add more element element to stack.\n");

}

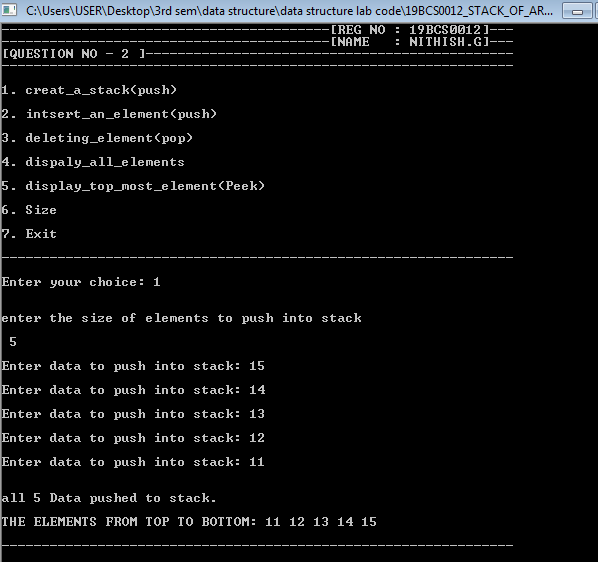
top++;

stack[top] = element;

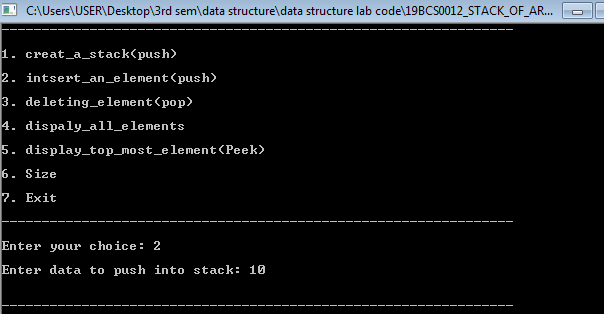
}

**Output**

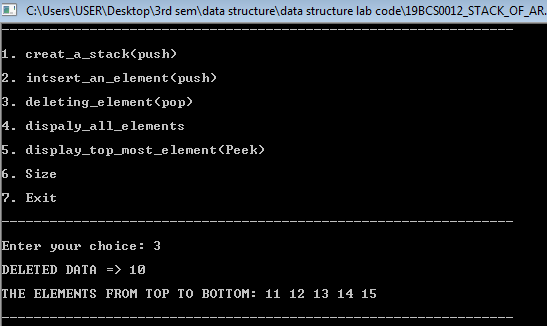
**A. create a stack**

****

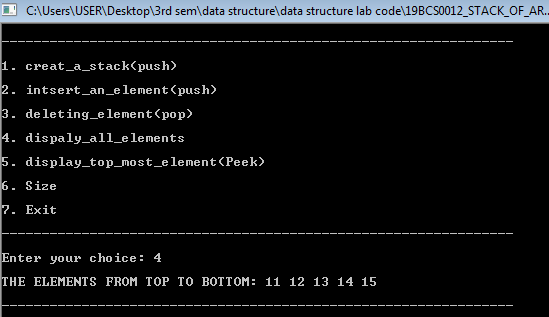
**b. insert an element**

****

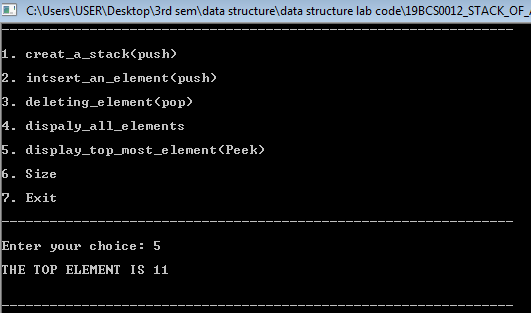
**c. deleting element**

****

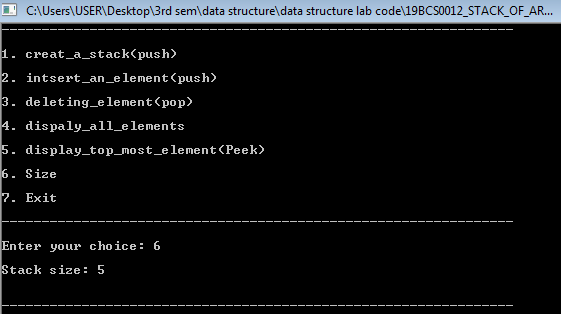
**d. display all elements**

****

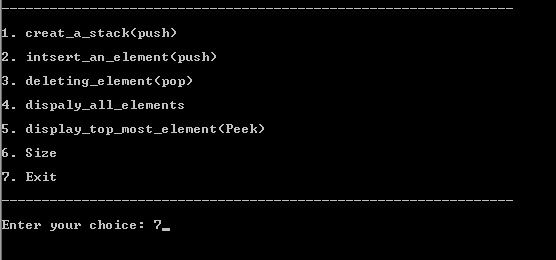
**e. display the topmost element**

****

**f. size**

****

**G. Exit.**

****

**3. Write a C program for the following algorithm to convert the infix expression to postfix expression.**

**Source code**

**#include<stdio.h>**

**#include<conio.h>**

**char stack[50];**

**int top = -1;**

**void push(char A);**

**char pop();**

**int priority(char B);**

**int main()**

**{**

**char T[50];**

**char \*Q, x;**

**printf("\n--------------------------------------[REG NO : 19BCS0012 ]-------\n");**

**printf("\n--------------------------------------[NAME : NITHISH.G ]-------\n");**

**printf("\n--[QUESTION NO.3]-------------------------------------------------\n\n\n");**

**printf("Enter the expression :: ");**

**scanf("%s",T);**

**printf("\n\n\t");**

**Q = T;**

**while(\*Q != '\0')**

**{**

**if(isalnum(\*Q))**

**printf("%c",\*Q);**

**else if(\*Q == '(')**

**push(\*Q);**

**else if(\*Q == ')')**

**{**

**while((x = pop()) != '(')**

**printf("%c", x);**

**}**

**else**

**{**

**while(priority(stack[top]) >= priority(\*Q))**

**printf("%c",pop());**

**push(\*Q);**

**}**

**Q++;**

**}**

**while(top != -1) {**

**printf("%c",pop());**

**}**

**getch();**

**return 0;**

**}**

**void push(char A)**

**{**

**top = top + 1;**

**stack[top] = A;**

**}**

**char pop()**

**{ if(top == -1)**

**return -1;**

**else**

**return stack[top--];**

**}**

**int priority(char B)**

**{**

**if( B == '(' )**

**return 0;**

**if(B == '+' || B == '-')**

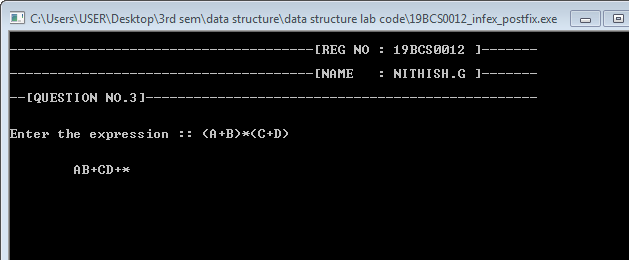
**return 1;**

**if(B == '\*' || B == '/')**

**return 2;**

**}**

**OUTPUT**

****