***NAME - AKRITI CHOUDHARY***

***ROLL NUMBER - 2005776***

***SUBJECT - DSA LAB***

***DATE - 5/10/2021***

***CLASS - B14***

***BRANCH - CSE***

**Question 1)Write a menu driven program to perform the following operations in a stack using array by**

**using suitable user defined functions for each case.**

**a) Check if the stack is empty**

**b) Display the contents of stack**

**c) Push**

**d) Pop**

**Verify & validate each function from main method.**

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int STACK[MAX], TOP;

void display(int stack[])

{

puts(" ");

int i = 0;

if (TOP == -1)

{

printf("Stack is Empty .\n");

return;

}

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

for (i = TOP - 1; i >= 0; i--)

{

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

}

puts(" ");

}

void PUSH(int stack[], int value)

{

if (TOP == MAX - 1)

{

printf("\nstack overflow\n");

return;

}

TOP++;

stack[TOP] = value;

}

void POP(int stack[])

{

int deletedItem;

if (TOP == -1)

{

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

return;

}

deletedItem = stack[TOP];

TOP--;

printf("deleted : %d\n", deletedItem);

return;

}

int main()

{

int value = 0;

int choice = 0;

TOP = -1;

while (choice != 4)

{

printf("Enter Choice :\n1 - display\n2 - PUSH\n3 - POP\n4 - Exit\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

display(STACK);

break;

case 2:

printf("Enter value to be insert :");

scanf("%d", &value);

PUSH(STACK, value);

break;

case 3:

POP(STACK);

break;

case 4:

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Program terminated\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

exit(0);

default:

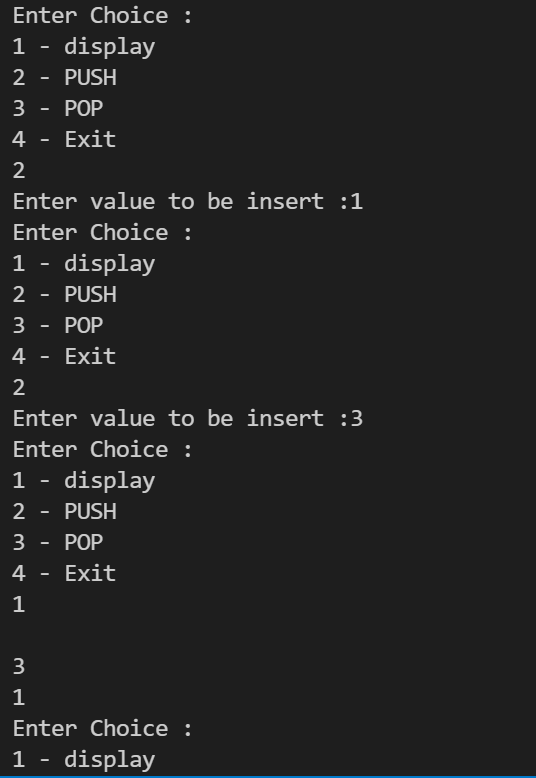
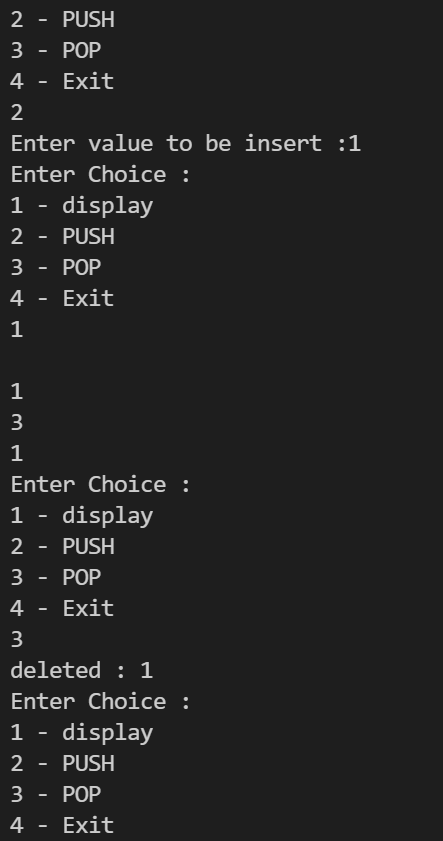
printf("\nInvalid choice.");

break;

}

}

}



**Question 2)** **Write a menu driven program to perform the following operations in a stack using dynamic array by**

**using suitable user defined functions for each case.**

**a) Check if the stack is empty**

**b) Display the contents of stack**

**c) Push**

**d) Pop**

**Verify & validate each function from main method.**

#include <stdio.h>

#include <stdlib.h>

struct Stack

{

int size;

int top;

int \*S;

};

void create(struct Stack \*st)

{

printf("Enter Size");

scanf("%d", &st->size);

st->top = -1;

st->S = (int \*)malloc(st->size \* sizeof(int));

}

void Display(struct Stack st)

{

int i;

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

printf("%d ", st.S[i]);

printf("\n");

}

void push(struct Stack \*st, int x)

{

if (st->top == st->size - 1)

printf("Stack overflow\n");

else

{

st->top++;

st->S[st->top] = x;

}

}

int pop(struct Stack \*st)

{

int x = -1;

if (st->top == -1)

printf("Stack Underflow\n");

else

{

x = st->S[st->top--];

}

return x;

}

int peek(struct Stack st, int index)

{

int x = -1;

if (st.top - index + 1 < 0)

printf("Invalid Index \n");

x = st.S[st.top - index + 1];

return x;

}

int isEmpty(struct Stack st)

{

if (st.top == -1)

return 1;

return 0;

}

int isFull(struct Stack st)

{

return st.top == st.size - 1;

}

int stackTop(struct Stack st)

{

if (!isEmpty(st))

return st.S[st.top];

return -1;

}

int main()

{

struct Stack st;

create(&st);

int choice = 0;

while (choice != 5)

{

puts("Enter choice :");

puts("1 - check if the stack is empty");

puts("2 - Display the content of the stack");

puts("3 - Push");

puts("4 - Pop");

puts("5 - Exit");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

int n = isEmpty(st);

if (n == 1)

{

puts("Stack is empty");

}

else

puts("Stack is not empty");

break;

}

case 2:

Display(st);

break;

case 3:

int x;

puts("Enter the value to be entered");

scanf("%d", &x);

push(&st, x);

break;

case 4:

pop(&st);

break;

case 5:

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Program terminated\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

break;

default:

puts("Invalid choice");

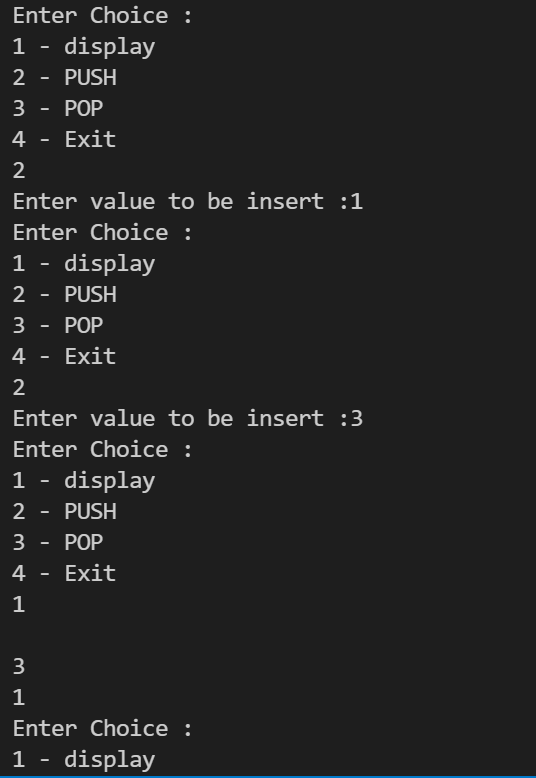
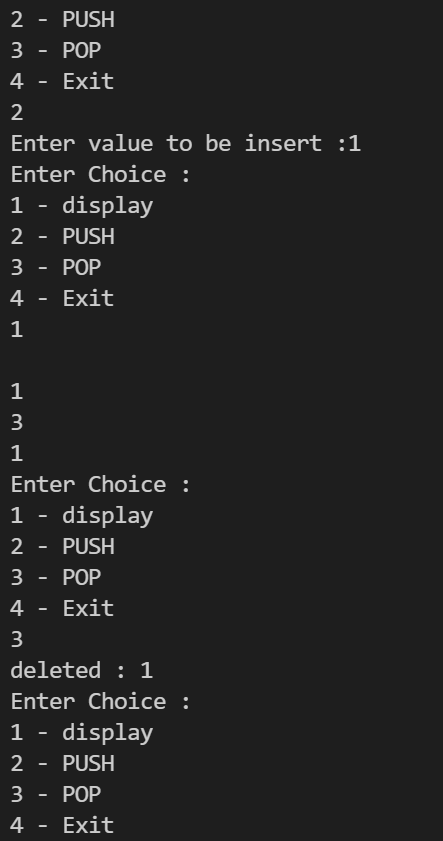
break;

}

}

return 0;

}



**Question 3)Write a menu driven program to perform the following operations in a stack linked list by**

**using suitable user defined functions for each case.**

**a) Check if the stack is empty**

**b) Display the contents of stack**

**c) Push**

**d) pop**

**Verify & validate each function from main method.**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void isEmpty(struct Node \*top)

{

if (top == NULL)

{

printf("Stack is empty\n");

}

else

{

printf("Stack not empty \n");

}

}

void traversal(struct Node \*ptr)

{

if (ptr == NULL)

{

printf("Stack underflow \n");

}

else

{

printf("Elements : \n");

while (ptr != NULL)

{

printf("%d \n", ptr->data);

ptr = ptr->next;

}

}

}

struct Node \*push(struct Node \*top, int num)

{

struct Node \*ptr = (struct Node \*)malloc(sizeof(struct Node));

ptr->data = num;

ptr->next = top;

top = ptr;

return top;

}

struct Node \*pop(struct Node \*top)

{

if (top == NULL)

{

printf("Stack underflow\n");

}

else

{

struct Node \*del;

del = top;

top = top->next;

printf("Element deleted = %d \n", del->data);

free(del);

}

return top;

}

int main()

{

int choice = 0, num;

struct Node \*top = NULL;

while (choice != 5)

{

printf("Enter the choice:\n");

printf("1 - Check if the stack is empty\n");

printf("2 - Display contents ( traverse )\n");

printf("3 - Push \n");

printf("4 - Pop operation \n");

printf("5 - Exit \n");

scanf("%d", &choice);

switch (choice)

{

case 1:

isEmpty(top);

break;

case 2:

traversal(top);

break;

case 3:

printf("Enter the number : \n");

scanf("%d", &num);

top = push(top, num);

break;

case 4:

top = pop(top);

break;

case 5:

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Program terminated\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

break;

default:

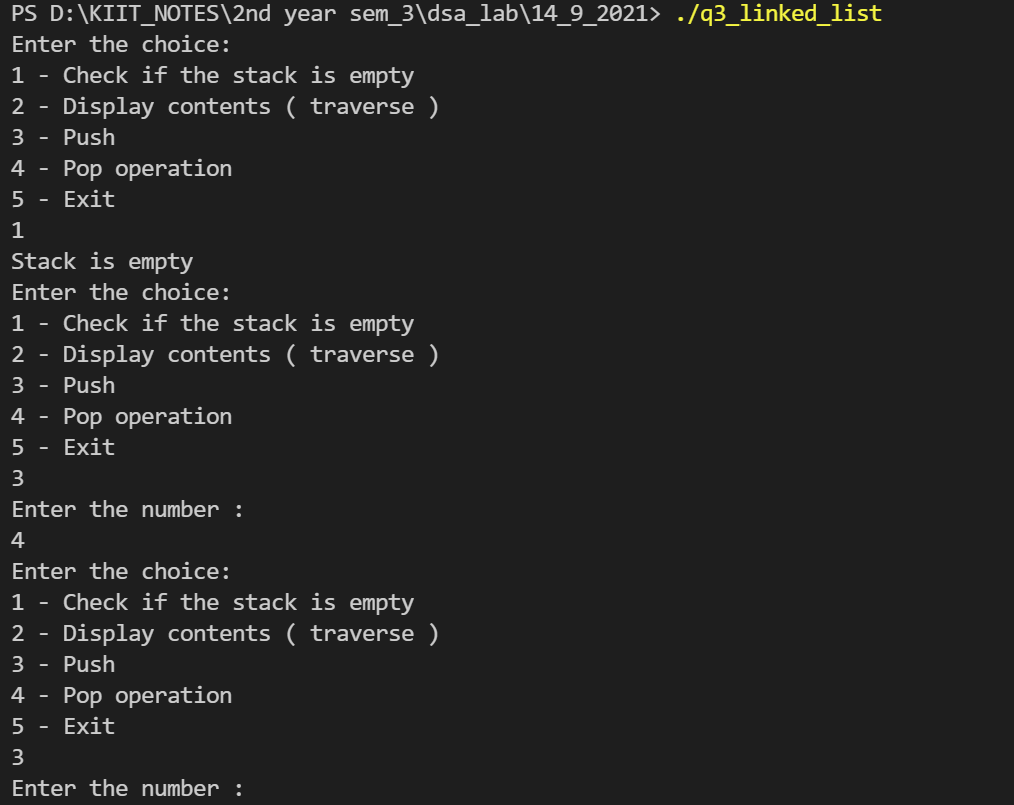
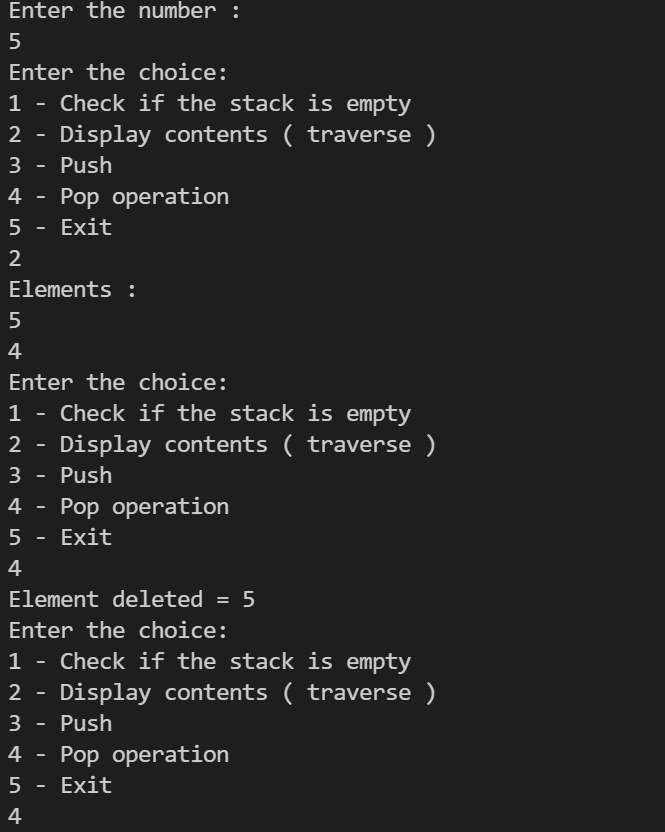
printf("Wrong input \n");

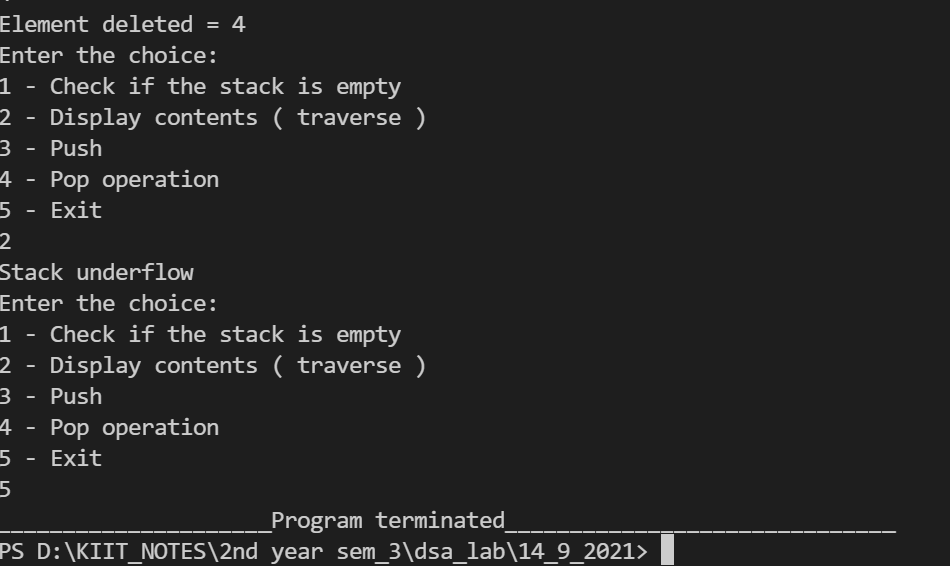
}

}

return 0;

}



**Question 4)WAP to convert an infix expression into its equivalent postfix notation.**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \* next;

};

typedef struct

{

struct Node \* top;

}Stack;

int IsEmpty(Stack s);

int push(Stack \* s,int v);

int pop(Stack \* s,struct Node \*\* v);

int IsOperand(int c);

int getVal(char a);

int IsLtoH(char a,char b);

int Infix\_Postfix(char \* inp , char \* out);

struct Node \* m;

int Infix\_Postfix(char \*inp , char \*out)

{

Stack s;

s.top=NULL;

int i=0,k=0,p,q;

while(inp[i] != '\0')

{

if(IsOperand(inp[i]))

{

out[k++]=inp[i];

}

else if(inp[i]==32)

{

i++;

continue;

}

else if(inp[i]=='(')

{

q=push(&s,inp[i]);

}

else if(inp[i]==')')

{

while(1>0)

{

p=pop(&s,&m);

if(p==1)

{

printf("Improper bracket pairs\n");

return 1;

}

if(m->data=='(')

break;

out[k++]=m->data;

}

}

else

{

if(IsEmpty(s))

{

q=push(&s,inp[i]);

}

else

{

p=pop(&s,&m);

if(p==1)

{

printf("Improper bracket pairs\n");

return 1;

}

if((m->data=='(')||(IsLtoH(m->data,inp[i])))

{

push(&s,m->data);

push(&s,inp[i]);

}

else

{

out[k++]=m->data;

i--;

}

}

}

i++;

}

while(!(IsEmpty(s)))

{

p=pop(&s,&m);

if(p==1)

{

printf("Improper bracket pairs\n");

return 1;

}

out[k++]=m->data;

}

out[k]='\0';

}

int IsLtoH(char a,char b)

{

if(getVal(a)<getVal(b))

return 1;

else

return 0;

}

int getVal(char a)

{

int t;

switch(a)

{

case '+':

case '-': t=1;

break;

case '\*':

case '/':t=2;

break;

case '^':t=3;

break;

}

return t;

}

int IsOperand(int c)

{

if( ((c>=65)&&(c<=90)) || ((c>=97)&&(c<=122)) )

{

return 1;

}

return 0;

}

int IsEmpty(Stack s)

{

if(s.top==NULL)

return 1;

return 0;

}

int push(Stack \*s,int v)

{

struct Node \*cur;

cur= (struct Node \*)malloc(sizeof(struct Node));

if(cur==NULL)

{

printf("Overflow");

return 1;

}

cur->data=v;

cur->next = s->top;

s->top=cur;

return 0;

}

int pop(Stack \* s,struct Node \*\* v)

{

if(IsEmpty(\* s))

{

printf("Underflow");

return 1;

}

\* v = s->top;

s->top=s->top->next;

return 0;

}

int main()

{

char inp[100];

int l=0,i=0;

printf("Enter infix expression:");

gets(inp);

while(inp[l] != '\0')

{

l++;

}

char out[l];

printf("The postfix Expression is:");

int c=Infix\_Postfix(inp,out);

while(out[i]!='\0')

{

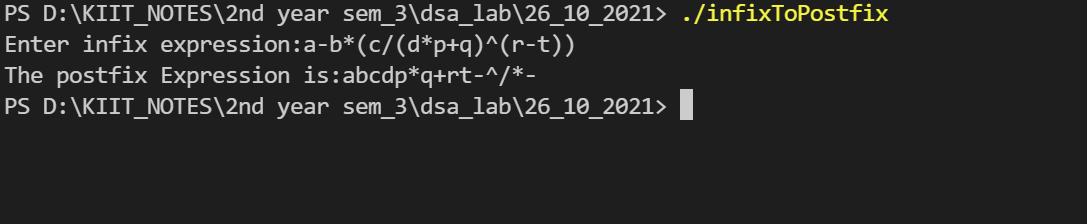
printf("%c",out[i]);

i++;

}

return 0;

}



**Question 5) WAP to convert an infix expression into its equivalent prefix notation.**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct Node

{

char data;

struct Node \* next;

};

typedef struct

{

struct Node \* top;

}Stack;

int IsEmpty(Stack s);

int push(Stack \* s,char v);

int pop(Stack \* s,struct Node \*\* v);

int IsOperand(int c);

int getVal(char a);

int IsHtoL(char a,char b);

int Infix\_Prefix(char \* inp , char \* out);

char\* reverse(char \*inout);

struct Node \* m;

int Infix\_Prefix(char \*inp , char \*out)

{

Stack s;

s.top=NULL;

int i=0,k=0,p,q;

inp=reverse(inp);

while(inp[i] != '\0')

{

if(IsOperand(inp[i]))

{

out[k++]=inp[i];

}

else if(inp[i]=='(')

{

q=push(&s,inp[i]);

}

else if(inp[i]==')')

{

while(1)

{

p=pop(&s,&m);

if(m->data=='(')

break;

out[k++]=m->data;

}

}

else

{

if(IsEmpty(s))

{

q=push(&s,inp[i]);

}

else

{

p=pop(&s,&m);

if((m->data=='('))

{

push(&s,m->data);

push(&s,inp[i]);

}

else if(IsHtoL(m->data,inp[i]))

{

out[k++]=m->data;

i--;

}

else

{

push(&s,m->data);

push(&s,inp[i]);

}

}

}

i++;

}

while(!(IsEmpty(s)))

{

p=pop(&s,&m);

out[k++]=m->data;

}

out[k]='\0';

out=reverse(out);

}

char\* reverse(char \*str)

{

int len=0;

while(str[len] != '\0')

{

len++;

}

char str\_tmp[len];

int j = 0;

for (int i = len - 1; i >= 0; i--)

{

if (\*(str+i) == '(')

{

\*(str\_tmp+j) = ')';

j++;

}

else if (\*(str+i) == ')')

{

\*(str\_tmp+j) = '(';

j++;

}

else

{

\*(str\_tmp+j) = \*(str+i);

j++;

}

}

\*(str\_tmp+(j++)) = '\0';

strcpy(str,str\_tmp);

return str;

}

int IsHtoL(char a,char b)

{

if(getVal(a)>getVal(b))

return 1;

else

return 0;

}

int getVal(char a)

{

int t;

switch(a)

{

case '+':

case '-': t=1;

break;

case '\*':

case '/':t=2;

break;

case '^':t=3;

break;

}

return t;

}

int IsOperand(int c)

{

if( ((c>=65)&&(c<=90)) || ((c>=97)&&(c<=122)) )

{

return 1;

}

return 0;

}

int IsEmpty(Stack s)

{

if(s.top==NULL)

return 1;

return 0;

}

int push(Stack \*s,char v)

{

struct Node \*cur;

cur= (struct Node \*)malloc(sizeof(struct Node));

if(cur==NULL)

{

printf("Overflow");

return 1;

}

cur->data=v;

cur->next = s->top;

s->top=cur;

return 0;

}

int pop(Stack \* s,struct Node \*\* v)

{

if(IsEmpty(\* s))

{

printf("Underflow");

return 1;

}

\* v = s->top;

s->top=s->top->next;

return 0;

}

int main()

{

char inp[100];

int l=0,i=0;

printf("Enter infix expression:");

scanf("%s",inp);

while(inp[l] != '\0')

{

l++;

}

char out[l];

printf("The prefix Expression is:");

int c=Infix\_Prefix(inp,out);

while(out[i]!='\0')

{

printf("%c",out[i]);

i++;

}

return 0;

}

