Practical-4

Stack Operations & Applications

1. Write a program to implement following stack operations using array with MAX elements. Use Character array

* PUSH
* POP
* PEEP
* CHANGE
* DISPLAY
* IS\_FULL
* IS\_EMPTY
* Code:

#include<stdio.h>

#include<stdlib.h>

#define size 10

void push(int);

void pop();

void display();

void peep(int);

void change(int,int);

int isFull();

int isEmpty();

int stack[size], top=-1;

void main()

{

int value,choice,pos,ans;

while(1)

{

printf("\n\nMenu..\n\n");

printf("1.Push\n2.Pop\n3.Display\n4.Peep\n5.Change\n6.Is full?\n7.Is empty?\n8.Exit");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch(choice)

{

case 1: printf("Enter the velue to be insert : ");

scanf("%d", &value);

push(value);

break;

case 2: pop();

break;

case 3: display();

break;

case 4: printf("Enter value to search: ");

scanf("%d", &value);

peep(value);

break;

case 5: printf("Enter value to be changed : ");

scanf("%d", &value);

printf("Enter position from top to change value : ");

scanf("%d", &pos);

change(value,pos);

break;

case 6: ans=isFull();

if(ans==1)

printf("\nStack is full");

else

printf("\nStack is not full");

break;

case 7: ans=isEmpty();

if(ans==1)

printf("\nStack is empty");

else

printf("\nStack is not empty");

break;

case 8: exit(0);

default:printf("\nInvalid choice!!");

}

}

}

void push(int value)

{

if(top==size-1)

printf("\nStack overflow.");

else

{

top++;

stack[top]=value;

printf("\nInsertion success");

}

}

void pop()

{

if(top==-1)

printf("\nStack underflow");

else

{

printf("\nDeleted : %d", stack[top]);

top--;

}

}

void peep(int value)

{

int found=0;

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

{

if(stack[i]==value)

{

printf("\nValue %d is at position %d from top",value,(top-i+1));

found=1;

}

if(found==0)

printf("\nValue is not present in stack..");

}

}

void change(int value,intpos)

{

int done=0;

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

{

if(pos==(top-i+1))

{

stack[top-pos+1]=value;

printf("\nValue changed");

done=1;

}

if(done==0)

printf("\nValue cannot be changed..");

}

}

int isFull()

{

if(top==size-1)

return 1;

else

return 0;

}

int isEmpty()

{

if(top==-1)

return 1;

else

return 0;

}

void display()

{

if(top==-1)

printf("\nStack is empty");

else

{

int i;

printf("\nStack elements are : \n");

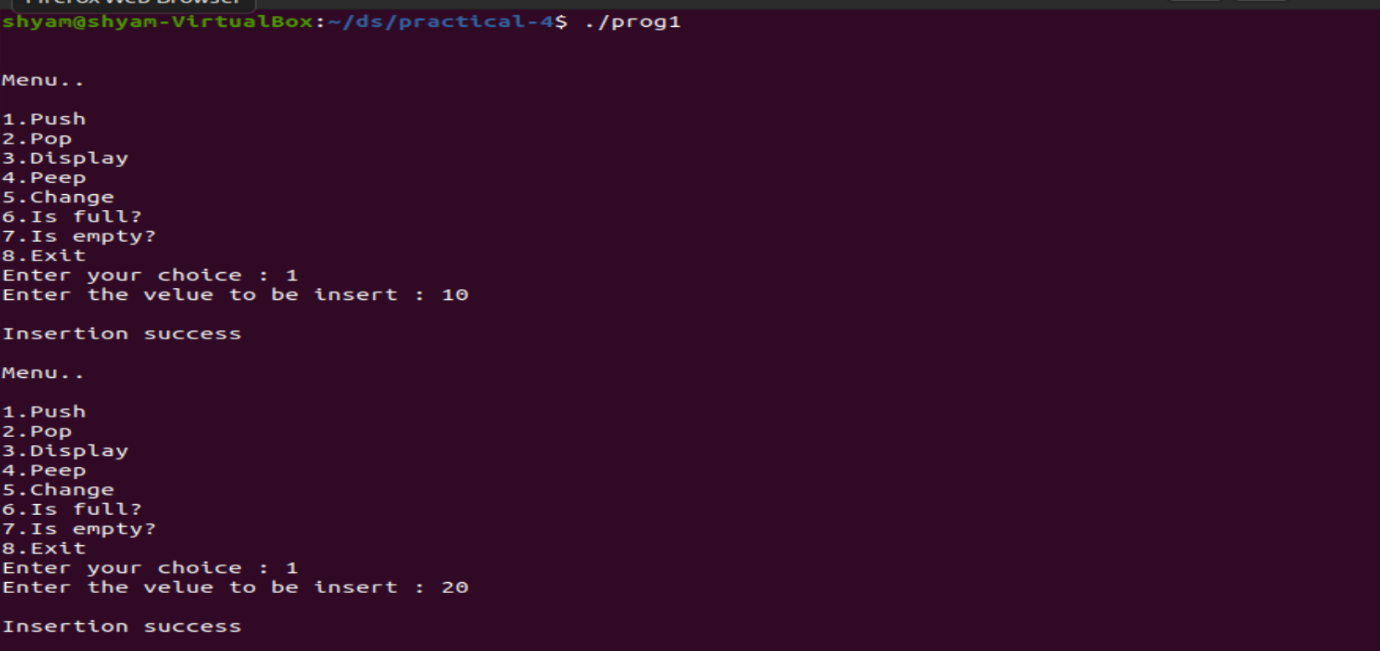
for(i=top;i>=0;i--)

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

}

}

* Output: 



1. Write a program to evaluate a postfix expression.

* Code:

#include<stdio.h>

#include<ctype.h>

int stack[25];

int top=-1;

void push(int item);

int pop();

void postfix(char postf[]);

int main()

{

int i;

char postf[25];

printf("Enter postfix expression for evaluation : ");

for(i=0;i<25-1;i++)

{

scanf("%c", &postf[i]);

if(postf[i]=='\n')

break;

}

postfix(postf);

return 0;

}

void push(int item)

{

if(top>=25-1)

printf("Overflow");

else

{

top++;

stack[top]=item;

}

}

int pop()

{

int item;

if(top<0)

printf("Underflow");

else

{

item=stack[top];

top--;

return item;

}

}

void postfix(char postf[])

{

int i,val,A,B;

char ch;

for(i=0;postf[i]!='\0';i++)

{

ch=postf[i];

if(isdigit(ch))

push(ch-'0');

else if(ch=='+'||ch=='-'||ch=='\*'||ch=='/')

{

A=pop();

B=pop();

switch(ch)

{

case '+':val=B+A;

break;

case '-':val=B-A;

break;

case '\*':val=B\*A;

break;

case '/':val=B/A;

break;

}

push(val);

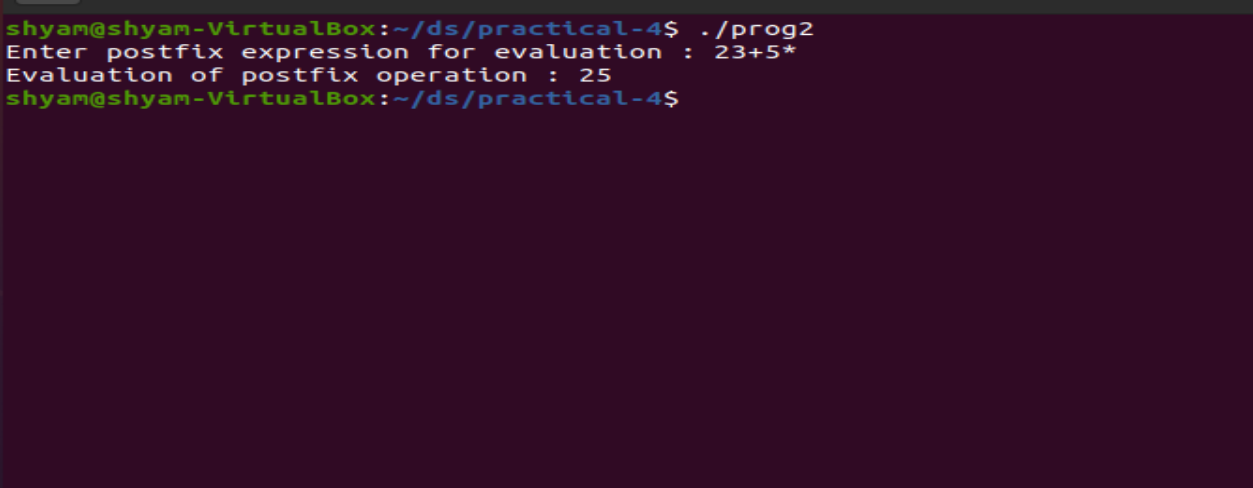
}

}

printf("Evaluation of postfix operation : %d\n", pop());

}

* Output:



1. Write a program for infix to postfix conversion of an expression.

* Code:

#include<stdio.h>

#include<ctype.h>

char stack[25];

int top=-1;

void push(char x)

{

stack[++top]=x;

}

char pop()

{

if(top==-1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x=='(')

return 0;

else if(x=='+'|| x=='-')

return 1;

else if(x=='\*'|| x=='/')

return 2;

return 0;

}

int main()

{

char exp[100];

char\*e,x;

printf("Enter the expression : ");

scanf("%s", exp);

printf("\n");

e=exp;

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

{

if(isalnum(\*e))

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

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

push(\*e);

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

{

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

printf("%c ",x);

}

else

{

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

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

push(\*e);

}

e++;

}

while(top!=-1)

{

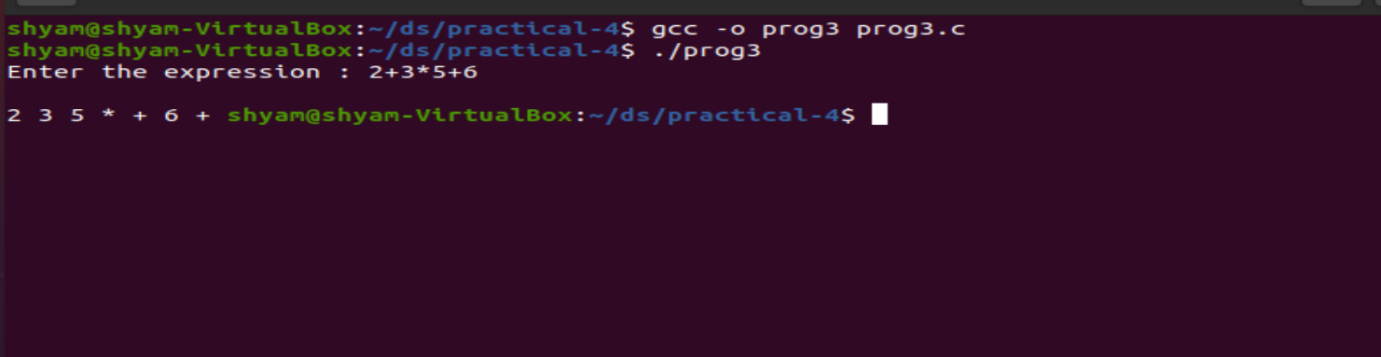
printf("%c ", pop());

}

return 0;

}

* Output:



1. Write a program to reverse the string using stack operations.

* Code:

#include<stdio.h>

#include<string.h>

#define size 50

int top,stackarr[size];

void push(char x)

{

if(top==size-1)

printf("Overflow");

else

stackarr[++top]=x;

}

void pop()

{

printf("%c", stackarr[top--]);

}

int main()

{

char str[size];

printf("Enter string : ");

gets(str);

int len=strlen(str);

int i;

printf("\nReverse string : ");

for(i=0;i<len;i++)

push(str[i]);

for(i=0;i<len;i++)

{

pop();

printf(" ");

}

return 0;

}

* Output:

