1) accept stack and perform push and pop operations.

#include<stdio.h>

#define MAX 5

struct stack

{

int a[MAX];

int top;

}s;

void init()

{

s.top==-1;

}

int isempty()

{

if(s.top==-1)

return 1;

else

return 0;

}

int isfull()

{

if(s.top==MAX-1)

return 1;

else

return 0;

}

void push(int num)

{

if(isfull())

{

printf("stack is full");

}

else

{

s.top++;

s.a[s.top]=num;

printf("insert..");

}

}

int pop()

{

int val;

if(isempty())

{

printf("stack is empty");

}

else

{

val=s.a[s.top];

s.top--;

}

return val;

}

void display()

{

int i;

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

{

printf("%d\n",s.a[i]);

}

}

int main()

{

int num,ch;

init();

do{

printf("\n 1:push \n 2:pop \n 3:display");

printf("\n enter choice=");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("enter num=");

scanf("%d",&num);

push(num);

break;

case 2:pop();

break;

case 3:display();

break;

}

}while(ch<4);

}

// 2) accept string and display reverse.

#include<stdio.h>

#define MAX 5

#include<process.h>

struct stack

{

char a[MAX];

int top;

}s;

void init()

{

s.top=-1;

}

int isfull()

{

if(s.top==MAX-1)

return 1;

else

return 0;

}

int isempty()

{

if(s.top==-1)

return 1;

else

return 0;

}

void push(char c)

{

if(isfull())

{

printf("stack is full");

}

else

{

s.top++;

s.a[s.top]=c;

}

}

void display()

{

int i;

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

{

printf("%c \n",s.a[i]);

}

}

int main()

{

int i;

char s1[20];

init();

printf("enter string=");

gets(s1);

display();

for(i=0;s1[i]!='\0';i++)

{

push(s1[i]);

}

printf("reverse string=");

display();

}

// 3) accept string and check pallindrome or not.

#include<stdio.h>

#define MAX 5

struct stack

{

char a[MAX];

int top;

}s;

void init()

{

s.top=-1;

}

int isfull()

{

if(s.top==MAX-1)

return 1;

else

return 0;

}

int isempty()

{

if(s.top==-1)

return 1;

else

return 0;

}

void push(char c)

{

if(isfull())

{

printf("stack is full");

}

else

{

s.top++;

s.a[s.top]=c;

}

}

char pop()

{

char val;

if(isempty())

{

printf("stack is empty");

}

else

{

val=s.a[s.top];

s.top--;

return val;

}

}

int main()

{

int i;

char s1[20];

printf("enter string=");

gets(s1);

init();

for(i=0;s1[i]!='\0';i++)

{

push(s1[i]);

}

for(i=0;s1[i]!='\0';i++)

{

if(s1[i]!=pop())

break;

}

if(isempty())

printf("string is pallindrome");

else

printf("string is not pallindrome");

}

// 4) accept expression and check is fully parenthesis or not.

#include<stdio.h>

#include<process.h>

# define MAX 5

struct stack

{

char a[MAX];

int top;

}s;

void init()

{

s.top=-1;

}

int isfull()

{

if(s.top==MAX-1)

return 1;

else

return 0;

}

int isempty()

{

if(s.top==-1)

return 1;

else

return 0;

}

void push(char c)

{

if(isfull())

{

printf("stack is full");

}

else

{

s.top++;

s.a[s.top]=c;

}

}

char pop()

{

char val;

if(isempty())

{

printf("stack is empty");

}

else

{

val=s.a[s.top];

s.top--;

}

}

int main()

{

int i;

char s1[100];

printf("entre expression=");

gets(s1);

init();

for(i=0;s1[i]!='\0';i++)

{

if(s1[i]=='(')

push(s1[i]);

if(s1[i]==')')

{

if(isempty())

{

printf("not parenthesis");

exit(0);

}

else

{

pop();

}

}

}

if(isempty())

printf("fully parenthesis");

else

printf("not fully parenthesis");

}

// 5) evaluate the expression using stack

#include<stdio.h>

#define MAX 50

struct stack // expression: A\*B$C+D\*E/F.

{

int a[MAX];

int top;

}s1;

void init()

{

s1.top=-1;

}

int isempty()

{

if(s1.top==-1)

return 1;

else

return 0;

}

int isfull()

{

if(s1.top==MAX-1)

return 1;

else

return 0;

}

void push(int num)

{

if(isfull())

{

printf("stack is full");

}

else

{

s1.top++;

s1.a[s1.top]=num;

}

}

int pop()

{

int v;

if(isempty())

{

printf("stack is empty");

}

else

{

v=s1.a[s1.top];

s1.top--;

return v;

}

}

int main()

{

char postfix[20];

int a,b,c,d,val1,val2,result,i;

printf("enter expression=");

gets(postfix);

printf("enter value of a,b,c,d=");

scanf("%d%d%d%d",&a,&b,&c,&d);

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

{

if(postfix[i]=='a')

push(a);

else if(postfix[i]=='b')

push(b);

else if(postfix[i]=='c')

push(c);

else if(postfix[i]=='d')

push(d);

else

{

val1=pop();

val2=pop();

if(postfix[i]=='+')

result=val2+val1;

if(postfix[i]=='-')

result=val2-val1;

if(postfix[i]=='\*')

result=val2\*val1;

if(postfix[i]=='/')

result=val2/val1;

push(result);

}

}

printf("result=%d",pop());

}