/\*Write a program to simulate the working of **stack using an array** with the following :

a) Push

b) Pop

c) to see the top element/peek

c) Display The program should print appropriate messages for stack overflow, stack underflow \*/

/\*

#include<stdio.h>

#include<stdlib.h>

#define SIZE 5

int top=-1,st[5];

void push();

void pop();

void display();

void top\_element();

void main()

{

  int ch;

  for(;;)

  {

    printf("\nStack Menu");

    printf("\n\n1.Push\n2.Pop\n3.Display\n4. to see the top element \n5.Exit");

    printf("\n\nEnter your choice: ");

    scanf("%d",&ch);

    switch(ch)

    {

      case 1: push();

          break;

      case 2: pop();

          break;

      case 3: display();

          break;

case 4: top\_element();

break;

      case 5: exit(0);

      default:

       printf("\nInvalid option!! Try again");

    }

  }

}

void push()

{

  int item;

  if(top==SIZE-1)

  {

    printf("\nStack Overflow\n");

  }

  else

  {

    printf("\nEnter element to be entered in stack:");

    scanf("%d",&item);

    top=top+1;

    st[top]=item;

  }

}

void pop()

{

  if(top==-1)

  {

    printf("\nStack Underflow\n");

  }

  else

  {

    printf("\nThe value deleted from stack is: %d",st[top]);

    top=top-1;

  }

}

void top\_element()

{

printf("the element at the top of the stack is-%d",st[top]);

}

void display()

{

  int i;

  if(top==-1)

  {

    printf("\nStack is empty\n");

  }

  else

  {

    printf("\nContents of the stack are: \n");

    for(i=top;i>=0;--i)

      printf("%d\n",st[i]);

  }

}

\*/

**/\* evaluation of postfix expression**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <ctype.h>

double s[40];

int top=-1;

double calc(char symbol,double op1,double op2){

switch(symbol){

case '+': return (op1+op2);

break;

case '-': return (op1-op2);

break;

case '\*': return (op1\*op2);

break;

case '/': return (op1/op2);

break;

case '$': return pow(op1,op2);

break;

}

}

void display(){

int j;

printf("Contents of stack updated\n");

for(j=0;j<=top;j++){

printf("%f\t",s[j]);

printf("\n");

}

}

int main(){

char exp[20];

double res;

printf("Enter expression\n");

scanf("%s",exp);

for(int i=0;i<strlen(exp);i++){

char symbol=exp[i];

if(isdigit(symbol)){

s[++top]=symbol-'0';

display();

}

else{

double op2=s[top--];

double op1=s[top--];

res=calc(symbol,op1,op2);

s[++top]=res;

display();

}

}

res=s[top];

printf("Result is : %f\n",res);

}

\*/

//Reversing a string

/\*

#include <stdio.h>

#include <string.h>

#define N 10

char string[N],revstring[N],s[N];

int top=-1,i,j;

int main(){

printf("Enter String\n");

scanf("%s",string);

for(i=0;i<strlen(string);i++){

s[++top]=string[i];

}

for(j=0;j<strlen(string);j++){

revstring[j]=s[top--];

}

printf("%s",revstring);

}

\*/

**//TO CHECK A PALINDROME STRING (only one word)**

/\*

#include <stdio.h>

#include <string.h>

#define N 10

char string[N],revstring[N],s[N];

int top=-1,i,j;

int main(){

printf("Enter String\n");

scanf("%s",string);

int flag;

for(i=0;i<strlen(string);i++){

s[++top]=string[i];

}

for(j=0;j<strlen(string);j++){

if(s[j]==s[strlen(string)-j-1]){

flag=1;

}

else{

flag=0;

break;

}

}

if(flag==1)

printf("Palindrome\n");

else

printf("Not palindrome\n");

}

\*/

//Infix to Postfix

/\*#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 40

char s[N],symbol;

int top=-1;

int F(char symbol){

switch(symbol){

case '+':

case '-': return 2;

case '\*':

case '/': return 4;

case '$':

case '^': return 5;

case '(': return 0;

case '#': return -1;

default: return 8;

}

}

int G(char symbol){

switch(symbol){

case '+':

case '-': return 1;

case '\*':

case '/': return 3;

case '$':

case '^': return 6;

case '(': return 9;

case ')': return 0;

default: return 7;

}

}

void infix\_postfix(char postfix[],char infix[]){

s[++top]='#';

int j=0;

for(int i=0;i<strlen(infix);i++){

symbol=infix[i];

while(F(s[top])>G(symbol)){

postfix[j++]=s[top--];

}

if(F(s[top])!=G(symbol)){

s[++top]=symbol;

}

else{

top--;

}

}

while(s[top]!='#'){

postfix[j++]=s[top--];

}

postfix[j]='\0';

}

int main(){

char postfix[20];

char infix[20];

int count1=0,count2=0;

printf("Enter Infix expression\n");

scanf("%s",infix);

for(int i=0;i<strlen(infix);i++){

if(infix[i]=='('){

count1++;

}

else if(infix[i]==')'){

count2++;

}

}

if(count1!=count2){

printf("Invalid Expression\n");

exit(0);

}

infix\_postfix(postfix,infix);

printf("%s",postfix);

}\*/

**/\***

**INFIX TO POSTFIX CONVERSION**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

char stack[20];

int top = -1;

int count1=0,count2=0;

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;

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

return 1;

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

return 2;

return 0;

}

int main()

{

int j=0;

char exp[100];

char res[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

for(int i=0;i<strlen(exp);i++)

{

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

count1++;

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

count2++;

}

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

{

if(isalnum(\*e))

res[j++]=\*e;

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

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

{

push(\*e);

}

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

{

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

res[j++]=x;

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

}

else

{

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

res[j++]=pop();

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

push(\*e);

}

e++;

}

while(top != -1)

{

res[j++]=pop();

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

}

if(count1 == count2)

printf("%s",res);

else {

printf("Inavalid expression\n");

exit(0);

}

return 0;

}

\*/

**/\* BALANCED PARENTHESIS**

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#define N 10

char s[N];

int top=-1;

void push(char symbol){

if(top==N-1){

printf("Overflow\n");

return;

}

s[++top]=symbol;

}

void pop()

{

if(top==-1){

printf("Underflow\n");

return;

}

top--;

}

int main(){

char expression[N];

printf("Enter expression:\n");

scanf("%s",expression);

if (strlen(expression)>10){

printf("10 symbols per sentance\n");

exit(0);

}

for(int i=0;i<strlen(expression);i++){

if(expression[i]=='('||expression[i]=='['||expression[i]=='{'){

push(expression[i]);

}

if(expression[i]==')'){

if(s[top]=='(')

pop();

else{

printf("Unbalanced\n");

exit(0);

}

}

else if(expression[i]==']'){

if(s[top]=='[')

pop();

else{

printf("Unbalanced\n");

exit(0);

}

}

else if(expression[i]=='}'){

if(s[top]=='{')

pop();

else{

printf("Unbalanced\n");

exit(0);

}

}

}

if(top==-1){

printf("Balanced\n");

}

else{

printf("Unbalanced\n");

}

}

\*/

**/\*factorial using recursion**

#include<stdio.h>

#include<stdlib.h>

int factorial(int n)

{

if(n<=1)

return 1;

return n \* factorial(n-1);

}

int main()

{

int n, ans;

printf("Enter the number\n");

scanf("%d", &n);

if(n>=0)

{

ans =factorial(n);

printf("%d\n", ans);

}

return 0;

}

\*/

#include<stdio.h>

void toh(int n,int A,int B,int C)

{

if(n>0)

{

toh(n-1,A,C,B);

printf("Move a disc from %d to %d\n",A,C);

toh(n-1,B,A,C);

}

}

int main()

{

int n;

printf("How many discs\n ");

scanf("%d",&n);

toh(n,1,2,3);

return 0;

}