//FOR FUTURE PROJECTS HOMIE YOU USE .h files now or create header.

|  |
| --- |
| 9 |
| 3 |
| 5 |

Data: 5 3 9 in stack ~~output: 9 3 5~~ but we want output to be 5 3 9

If we want it to come in orde thene push it to another stack

Then pop it out again and youll get that order.

atoyota

-------🡪 //if a phrase reads the same from both directions, we call it

A Toyota // PALINDROME

🡨------

Atoyota

How do we collect these from left to right and right to left?

|  |
| --- |
| a |
| t |
| o |
| y |
| o |
| T |
| A |

|  |
| --- |
| a |
| t |
| o |
| y |
| o |
| T |
| A |

RtoL LtoR

//we want these to be reversed to each other

//what we want to do is create a temporary stack

|  |
| --- |
| ~~a~~ |
| ~~t~~ |
| ~~o~~ |
| ~~y~~ |
| ~~o~~ |
| ~~T~~ |
| ~~A~~ |

|  |
| --- |
| A |
| T |
| o |
| y |
| o |
| t |
| a |

🡨Temporary stack //then pop it again

L to R

//STACK LR, RL, Temp

STACK<char, 80> LR, RL, Temp

LR.clearStack();

RL.clearStack();

Temp.clearStack(); //set the counter to all 3 stacks to zero

//ask the user to enter a sentence

cout<<”Enter a sentence. . .”;

//..if it is upper case letter push it to stack RL and Temp

char c; //we want to not read spaces frm user n stuff

while(cin.get(c), c!=’n’)

{

if(isalpha(c))

{

//change it to upper or lower first

c=tolower(c);

RL.pushStack(c);

Temp.pushStack(c);

}//end if

}//end while

|  |
| --- |
| a |
| t |
| o |
| y |
| o |
| t |
| a |

//copy temp into LR

while(!temp.emptyStack())

{

. . . pop temp and put it in LR 🡨 LR

}

//test for palindrome!!

while(!RL.emptyStack())

{

char c1=RL.popStack();

char c2=LR.popStack();

if(c1!=c2) break;

//if the stack is empty then it is a palindrome and everything matched

}

if(RL.emptyStack()==true)

{

cout<<”u have palindrome”<<endl;

}

else

cout<<”not palindrome”<<endl;

Prefix­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_infix\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_postfix

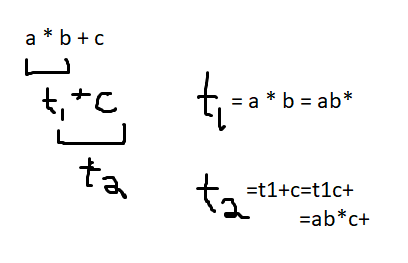
+a b a + b a b+

-a b a – b a b-

🡨 a \* b + c 🡪

//to convert a \* b + c to postfix and prefix

a \* b + c



t1=a\*b=Xab

t2=t1+c=+t1c=+\*ab

//so

+\*abc 🡨 a\*b+c 🡪 ab\*c-

example if we have something like this

x=a+b\*c;

//the technique the compiler uses.. to find solution

infix= operation is in the middle

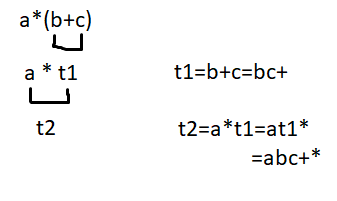
//for adition a b are the same and it doesn’t matter

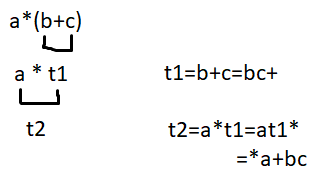
//ANOTHER EXAMPLE

Prefix­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_infix\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_postfix

a\*(b+c)

convert a\*(b+c) to postfix and prefix



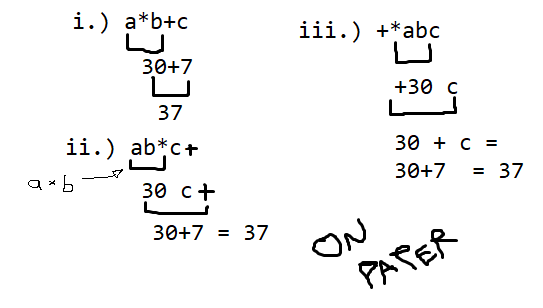


//so we have

\*a+bc a\*(b+c) abc+\*

Find the value of the following

for a=5, b=6, c=7



//now to do it on camputor version!

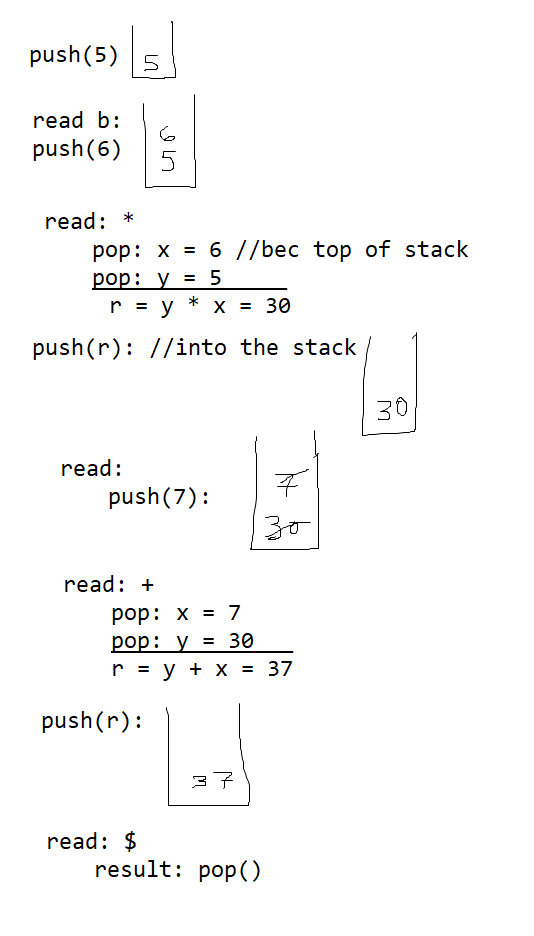
Enter a postfix expression: ab\*c+ suppose

a=5, b=6, c=7;

//the compiler puts a $ sign at the end to mark end of expression

like this ab\*c+$

read: a //push the value of a in a stack



STACK <int, 10> S;

int a=5, b=6, c = 7, r//result; char p;

cout<<”Enter a postfix expression:”;

cin>>p;

//pmuch creating a stack that does programming MATH with post and prefix

while(p!=’$’)

{

switch(p)

{

case(‘+’) : x=S.popStack();

y=s.popStack();

r=y+x;

S.pushStack(r);

break;

case(‘-‘) etc. . .

}

cin>>p; //bec we wanted to. . .

}

cout<<”Value of expression=”<<S.popStack();