//Assignment no 4

#include <iostream>

#include <string.h>

#define max 10

using namespace std;

class node

{

public:

char data;

node \*left;

node \*right;

};

class stack

{

node \*stk[max];

int top;

public:

stack()

{

top = -1;

}

int isempty();

int isfull();

void push(node \*x);

node\* pop();

void display();

};

class expr

{

node\* root;

public:

expr()

{

root = NULL;

}

node \* getroot()

{

return(root);

}

void createexpr();

void inorder(node\* root);

};

int stack :: isempty()

{

if(top == -1)

{

return 1;

}

else

{

return 0;

}

}

int stack :: isfull()

{

if(top == max-1)

{

return 1;

}

else

{

return 0;

}

}

void stack :: push(node \*x)

{

int c = isfull();

if(c != 1)

{

stk[++top] = x;

}

else

{

cout << "\nStack is full";

}

}

node\* stack :: pop()

{

node \*t;

int j = isempty();

if(j != 1)

{

t = stk[top];

return(stk[top--]);

}

else

{

return(NULL);

}

}

void stack :: display()

{

for(int i=0; i<=top; i++)

{

cout << stk[i]->data << " ";

}

}

void expr :: createexpr()

{

node\* new1;

char str[15];

int length;

stack s;

cout << "\nEnter a prefix expression: ";

cin >> str;

length = strlen(str);

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

{

new1 = new node;

new1->data = str[i];

new1->left = NULL;

new1->right = NULL;

if(new1->data == '+' || new1->data == '-' || new1->data == '\*' || new1->data == '/' || new1->data == '%')

{

new1->left = s.pop();

new1->right = s.pop();

s.push(new1);

}

else

{

s.push(new1);

}

}

root = s.pop();

}

void expr :: inorder(node\* root)

{

if(root != NULL)

{

inorder(root->left);

cout << root->data;

inorder(root->right);

}

}

int main()

{

int ch;

stack g;

expr e;

do

{

cout << "\n--------MENU-------";

cout << "\n1. Push";

cout << "\n2. Pop";

cout << "\n3. Display";

cout << "\n4. Expression Tree";

cout << "\n5. Exit";

cout << "\nEnter your choice: ";

cin >> ch;

switch(ch)

{

case 1:

node \*nn;

nn = new node();

cout << "\nEnter element to push: ";

cin >> nn ->data;

g.push(nn);

break;

case 2:

node \*t;

t = new node();

t = g.pop();

cout << "\nElement has been popped"<<t->data;

break;

case 3:

g.display();

break;

case 4:

e.createexpr();

e.inorder(e.getroot());

break;

case 5:

exit(0);

break;

default:

cout << "\nInvalid choice entered";

}

}while(ch != 5);

return 0;

}

output:

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ g++ expr\_tree.cpp

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ ./a.out

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 1

Enter element to push: 1

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 1

Enter element to push: 2

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 2

Element has been popped2

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 3

1

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 4

Enter a prefix expression: +--a\*bc/def

a-b\*c-d/e

--------MENU-------

1. Push
2. Pop
3. Display
4. Expression Tree
5. Exit

Enter your choice: 5

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$