Test1.txt:

23.8 4 \* 19.5 2 - +

6.8 10.5 -

36 6 /

Output:

23.8 4 \* 19.5 2 - + is 112.7

6.8 10.5 - is -3.7

36 6 / is 6

Test2.txt

10 6 \* 3 / 5 4 + -

5 5 5 5 5 10 + - + \* /

3 5.5 - 7.6 +

12 1.5 /

Output:

10 6 \* 3 / 5 4 + - is 11

5 5 5 5 5 10 + - + \* / is -0.2

3 5.5 - 7.6 + is 5.1

12 1.5 / is 8

Test3.txt

0.01 4 - 3 \*

4 4 4 - +

2 3 4.3 \* \*

Output:

0.01 4 - 3 \* is -11.97

4 4 4 - + is 4

2 3 4.3 \* \* is 25.8

**yourProgram.cpp:**

#include "Stack.h"

#import <string>

#import <vector>

#include <fstream>

#include <sstream>

#include <iostream>

#include <assert.h>

using namespace std;

ifstream getFileInput();

//input: user input

//output: ifstream of file

void runFileCommands(ifstream& fileInput,vector<string>& sarray,vector<string>& darray);

//input: stream fileInput, vector holding input string lines, vector holding answer

//output: none

//side effects: responsible for updating the output file based the input file it was given

void updateVector(string& line,vector<string> & sarray,vector<string>& darray);

//input: line from input file, array holding all lines from input,

//line holding the answer

//output: none

//side effects:breaks down each line of the file into the 2 arrays

void outputfile(vector<string> & sarray,vector<string>& darray);

//input: two string vectors holding each line of input file and the answer to each lines

//output:none

//side effects: Creates an output file with the output desired

string evaluate(string line);

//input:string line

//output: String answer to the line of input given

//side effects: Responsible for computing the numbers in the correct way

int main(){

vector<string> stringArr;

vector<string> doubleArr;

// cout<<"Main started"<<endl;

ifstream myfile(getFileInput());

// cout<<"My file opened correctly"<<endl;

runFileCommands(myfile,stringArr,doubleArr);

return 0;

}

ifstream getFileInput(){

cout<<"Enter file name: "<<endl;

string test;

getline(cin,test);

ifstream myfile(test);

return myfile;

}

void runFileCommands(ifstream& fileInput,vector<string>& sarray,vector<string>& darray){

// cout<<"Run file commands started"<<endl;

string line;

if(fileInput.is\_open()){

while(getline(fileInput,line)){

updateVector(line,sarray,darray);

}

outputfile(sarray,darray);

}else{

cout<<"Failed to open file"<<endl;

}

}

void updateVector(string& line,vector<string> & sarray,vector<string>& darray){

//cout<<"updateVector started"<<endl;

sarray.push\_back(line);

// cout<<"This is evaluate line"<<endl;

string tester = evaluate(line);

// cout<<tester<<endl;

darray.push\_back(tester);

}

void outputfile(vector<string> & sarray,vector<string>& darray){

//cout<<"Output file "<<endl;

ofstream out;

out.open("output.txt");

for(int i =0;i<sarray.size();i++){

string words;

istringstream iss(sarray.at(i));

while(iss>>words){

out<<words<<" ";

}

out<<"is "<<darray.at(i)<<endl;

}

out.close();

cout<<"Output file created"<<endl;

}

string evaluate(string line){

// cout<<"evaluate started"<<endl;

Stack numbers;

string subs;

istringstream iss(line);

while(iss>>subs){

if(isdigit(subs.at(0))){

numbers.Push(subs);

}else{

// cout<<"Operand: "<<subs<<endl;

string operand = subs;

double top = -1000.0;

double second = -1000.0;

istringstream streams(numbers.Pop());

streams>>top;

istringstream streamer(numbers.Pop());

streamer>>second;

// cout<<second<<" "<<top<<endl;

if(operand=="\*"){

ostringstream streamed;

streamed<<(top\*second);

// cout<<" \* "<<endl;

numbers.Push(streamed.str());

}else if(operand=="-"){

ostringstream streamed;

streamed<<(second-top);

// cout<<" - "<<endl;

numbers.Push(streamed.str());

}else if(operand=="+"){

ostringstream streamed;

streamed<<(top+second);

// cout<<" + "<<endl;

numbers.Push(streamed.str());

}else if(operand =="/"){

ostringstream streamed;

streamed<<(second/top);

// cout<<" / "<<endl;

numbers.Push(streamed.str());

}

}

}

return numbers.Pop();

}

**Stack.cpp**

#include "Stack.h"

#include <string>

#include <iostream>

#include <assert.h>

using namespace std;

Stack::Stack(){

head = nullptr;

length = 0;

}

bool Stack::isEmpty() const{

return (length == 0);

}

int Stack::getLength() const{

return length;

}

bool Stack::Push(const string & newEntry){

node\* p = new node;

p->data = newEntry;

p->next = head;

head = p;

length++;

// cout<<"Stack function worked"<<endl;

return true;

}

string Stack::Pop(){

assert(length!=0);

node\*b = head;

node\*c = head->next;

string str = head->data;

head=c;

delete b;

length--;

// cout<<"Stack popped worked"<<endl;

return str;

}

string Stack::Peek() const{

assert(length!=0);

return head->data;

}

Stack::~Stack(){

node\* a = head;

while(a!=nullptr){

node \*b = a->next;

delete a;

a = b;

}

// cout<<"Stack deleted"<<endl;

}

**Stack.h**

#ifndef Stack\_

#define Stack\_

#import <string>

using namespace std;

class Stack {

public:

Stack();

//Constructor

~Stack();

//Destructor

bool isEmpty() const;

//returns true if stack is empty, otherwise returns false

int getLength() const;

//returns length of the Stack

bool Push(const string & newEntry);

//Adds a node wtih string data type to the Stack

string Pop();

//Precondition: Must have atleast one node in the Stack

//Returns the top of the stack

string Peek() const;

//Precondition: Must have atleast one node in the stack

private:

class node{

public:

node \*next;

string data;

};

int length;

node \*head;

};

#endif