// Shraddha Jamadade

// Student Id: 110287963

// CSci 117 ASsignment 3

// 09/21/2017

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//

// Interpreter for language FresnoF17 with +,-,\*, /, (), ^ operations, variable declaration, assignment statement, print statement

// Input is the data file with FresnoF17 program, spaces are allowed.

// There are two program files- Assign1.cpp and Assign2.cpp

// Assign1.cpp has the code to evaluate the expressions as per grammar rules.

// Assign2.cpp contains the code for interpreter for the language FresnoF17 to check the correctness of the language given in input data file.

// Program- Assign2.cpp uses functions-

// int Exp(),Term(),Exp2(int),Term2(int),Fact(),Num() functions from //ASsign1.cpp

// compile: $>g++ Assign1.cpp Assign2.cpp

// run with: $>./a.out FresnoF17

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// CODE

// Assign1.cpp

#include <cstdlib> //for atoi()

#include <stdio.h>

#include <stdlib.h>

#include<math.h> //for mathematical operators (power)

#include <fstream>

using namespace std;

//input file stream defined in file prog3.cpp

extern ifstream fin;

int Exp(),Term(),Exp2(int),Term2(int),Fact(),Num();

//Exp() function calls the Exp2() function for solving addition and subtraction operations

int Exp()

{

return Exp2(Term()); //Result of Fact() function is passed as parameter to Term2()

}

//Term() function calls the Term2() function for solving multiplication and division operations

int Term()

{

return Term2(Fact()); //Result of Term() function is passed as parameter to Exp2()

}

//Exp2() recursive function where the parameter passed is the result of multiplication and division operations

//checks whether it is end of expression

//gets one character from input expression

//skips blank spaces if any

//checks for addition'+' and subtraction'-' operations

//performs operation

//returns result

int Exp2(int inp)

{

int result=inp;

char a;

if((a=getchar()) != '\n'){

if(a == '+')

result = Exp2(result + Term());

else if(a == '-')

result = Exp2(result - Term());

}

return result;

}

//Term2() recursive function where the parameter passed is the result of the power operation

//checks whether it is end of expression

//gets one character from input expression

//skips blank spaces if any

//checks for multiplication'\*' and division'/' operations

//performs operation

//returns result

int Term2(int inp)

{

int result = inp;

char a;

fin.read(&a,1);

if(a != '\n'){

if(a == '\*')

result = Term2(result\*Fact());

else if(a == '/')

result = Term2(result/Fact());

else if (a == '+' || a == '-' || a == ')')

fin.unget();

}

if(a == '\n')

fin.unget();

return result;

}

//Fact() function for performing power operations

//checks whether it is end of expression

//gets one character from input expression

//skips blank spaces if any

//checks for power'^' operations

//performs operation

//returns result

int Fact()

{

int result = Num();

char a;

fin.read(&a,1);

if(a != '\n')

{

if(a == '^')

result = pow((float)result,(float)Fact());

else

fin.unget();

}

if(a == '\n')

fin.unget();

return result;

}

//Num() recursive function

//gets one character from input expression

//skips blank spaces if any

//checks for () operations

//converts each character into numeric number

//returns result

int Num()

{

char a[2];

int result;

int resultExp;

a[0] = getchar();

// ignore blank spaces in input string

while(a[0] == ' '){

a[0] = getchar();

}

a[1] = '\0';

if ( a[0] == '('){

resultExp = Exp();

return resultExp;

}

result = atoi(a);

return result;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Assign2.cpp

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

using namespace std;

//Exp function used to evaluate an expression is defined in file prog2.cpp

extern int Exp();

int indexx = 0; //global index for program string

int sym\_index = 0; //global for symbol table index

//input file stream handler

ifstream fin;

//vector to store the list of known keywords in the language defined by the given //grammar

vector<string> keywords(6);

//creates an array of symbol table where each node stores an identifier name, data //type and its value

struct node{

char id; //single letter var name

string type;

double val;

}sym\_table[100]; //symbol table with 100 entries

string read\_word()

{

string word;

if(fin.is\_open()) //open data file

{

fin >> word;

}

return word;

}

void declaration(string type) //para is type (int or double)

{

char id;

bool flag = false;

while(fin.read(&id,1))

{

if(id == ' ' || id == ',')

continue;

else if(id == ';'){

flag = true;

break;

}

else if(id == '\n')

break;

if(isalpha(id)){

sym\_table[indexx].id =(char)id;

sym\_table[indexx].type = type;

indexx++;

}else{

cout<<"\nSYNTAX ERROR : identifier name not an alphabet for identifier "<<id<<"\n";

fin.close();

exit(1);

}

}

if(!flag){

cout<<"\nSYNTAX ERROR : Declaration statement not terminated with ;\n";

fin.close();

exit(1);

}

}

void declarations() //recursively calls declaration()

{

string word = read\_word();

if(word == "begin"){

return;

}

else if (word == "int" || word =="double")

{

declaration(word); //for one line of declaration until ';'

}

declarations(); //recursion for next declaration

}

void print\_stmt()

{

int index = -1;

bool flag = false;

char id ;

char var;

fin.read(&id,1);

while(id == ' '){

fin.read(&id,1);

}

if(isalpha(id)){

var =id;

fin.read(&id,1);

if(id != ';'){

cout<<"\nSYNTAX ERROR : print statement not terminated with ;\n";

fin.close();

exit(1);

}

for(int i =0;i<indexx;i++){

if(var == sym\_table[i].id){

flag = true;

index = i;

break;

}else{

continue;

}

}

if(!flag){

cout<<"\nSEMANTIC ERROR : identifier "<<var<<" in Print Statement not Declared \n";

fin.close();

exit(1);

}else{

cout<<endl;

cout<<sym\_table[index].val;

}

}else{

fin.unget();

cout<<endl;

cout<<Exp();

cout<<endl;

}

}

void assign(string id)

{

char c;

fin.read(&c,1);

int resultExp;

bool flag = false;

int index=-1;

while(c == ' '){

fin.read(&c,1);

}

//look for id in the symbol table if it

if(!isalpha(id[0])){

cout<<"\nSYNTAX ERROR : target identifier "<<id[0]<<" not an alphabet in assignment statement\n";

fin.close();

exit(1);

}

for(int i =0;i<indexx;i++){

if(id[0] == sym\_table[i].id){

flag = true;

index = i;

break;

}else{

continue;

}

}

if(flag != true){

cout<<"\nSEMANTIC ERROR : identifier "<<id[0]<<" in assignment statement not Declared \n";

fin.close();

exit(1);

return;

}

if (c == '='){

if( sym\_table[index].type != "int"){

cout<<"\nSEMANTIC ERROR : Type mismatch in assignment statement, identifier "<<sym\_table[index].id<<" of type "<<sym\_table[index].type<<" but expecting int type\n";

exit(1);

return;

}

resultExp = Exp();

sym\_table[index].val = resultExp;

}

else{

cout<<"\nSYNTAX ERROR : in Assignment statement\n";

return;

}

}

void statement(string word)

{

if(word == "print"){

print\_stmt();

}

else if (word.length()==1 && isalpha(word.at(0))) //a = c; or a = 2+3\*5;

{

assign(word);

}

}

void statements()

{

string word = read\_word();

if(word == "end" ){

return ;

}

else

statement(word);

statements();

}

void initKeywrdTbl()

{

keywords[0]="program";

keywords[1]="begin";

keywords[2]="end";

keywords[3]="int";

keywords[4]="double";

keywords[5]="print";

}

void parseForLexicalErr()

{

string word;

bool found = false;

bool begFlag = false;

fin.open("FresnoF17.txt");

while(fin >> word){

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

if(word == keywords[i]){

found = true;

break;

}

else

found = false;

}

if(found == true || ((word.size()<3) && isalpha(\*(word.begin()))) || word == "=" || (\*(word.begin())) == '(' || isdigit(\*(word.begin())) ){

continue;

}

else{

cout<<endl;

cout<<"\nLEXICAL ERROR: Keyword "<<word<<" not valid\n";

cout<<endl;

fin.close();

exit(1);

}

}

fin.close();

fin.open("FresnoF17.txt");

fin.seekg(0,ios\_base::beg);

fin >> word;

if(word != "program"){

cout<<"\nSYNTAX ERROR : Keyword PROGRAM not present at begining of the program\n";

fin.close();

exit(1);

}

bool endFlag = false;

while(fin >> word){

if(word.compare(0,5,"begin") == 0)

begFlag=true;

if((word == "int" || word == "double") && begFlag){

cout<<"\nSYNTAX ERROR : Declarations found even after BEGIN Keyword\n";

fin.close();

exit(1);

}

if(word.compare(0,3,"end") == 0){

endFlag = true;

continue;

}

if((word == "print" || word == "=" || isalpha(\*(word.begin())) ) && endFlag){

cout<<"\nSYNTAX ERROR : Statements found even after END Keyword\n";

fin.close();

exit(1);

}

}

if(!begFlag){

cout<<"\nSYNTAX ERROR :BEGIN Keyword not found in the program\n";

fin.close();

exit(1);

}

if(!endFlag){

cout<<"\nSYNTAX ERROR :END Keyword not found in the program\n";

fin.close();

exit(1);

}

fin.close();

}

int main()

{

string word;

initKeywrdTbl();

parseForLexicalErr();

fin.open("FresnoF17.txt");

fin.seekg(0,ios\_base::beg);

word = read\_word();

if(word == "program")

{

declarations();

// to display symbol\_table entries

// cout<<"++symbol table++"<<endl;

// for(int i=0; i<sym\_index; i++)

// cout<<sym\_table[i].id<<" "<<sym\_table[i].type<<" "<<sym\_table[i].value<<endl;

statements();

}

else{

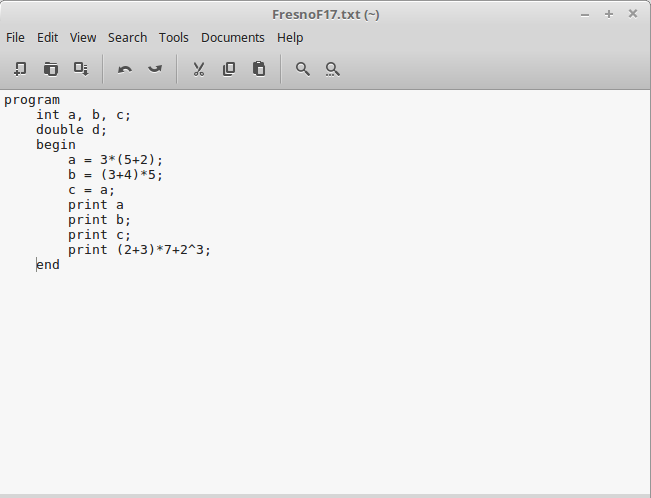
cout<<"\nSYNTAX ERROR : Keyword PROGRAM not present at begining of the program\n";

}

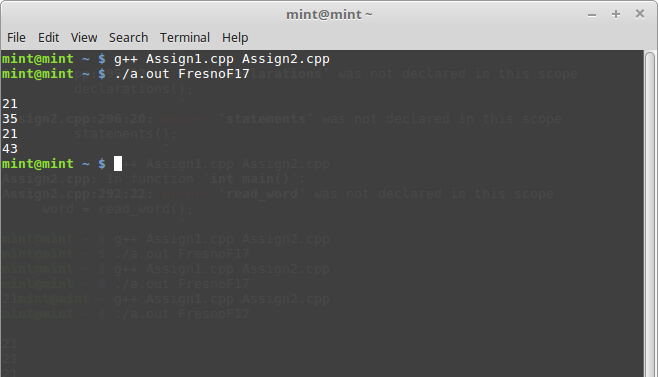
}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input data file which contains the sample FresnoF17 program:

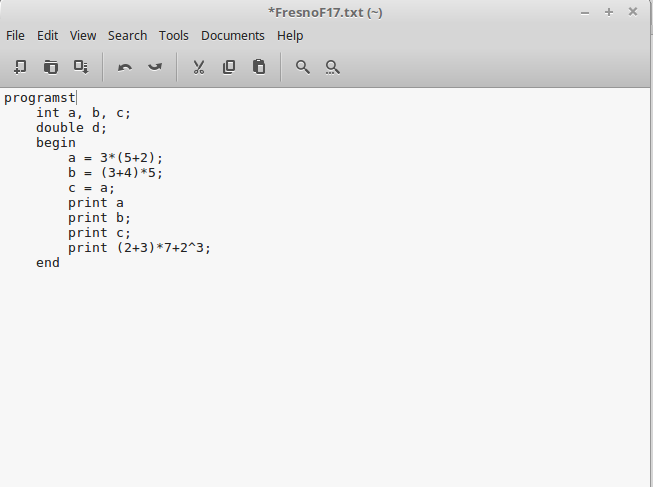


Output

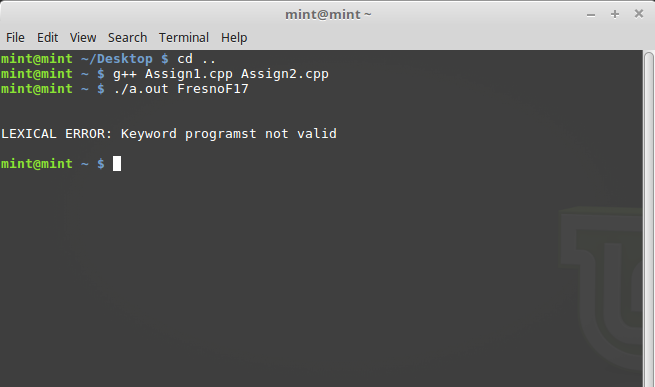


Lexical, Syntax and Semantic Error

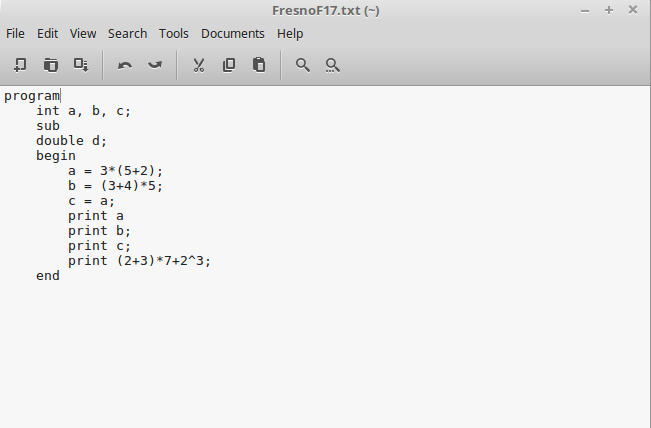
Input



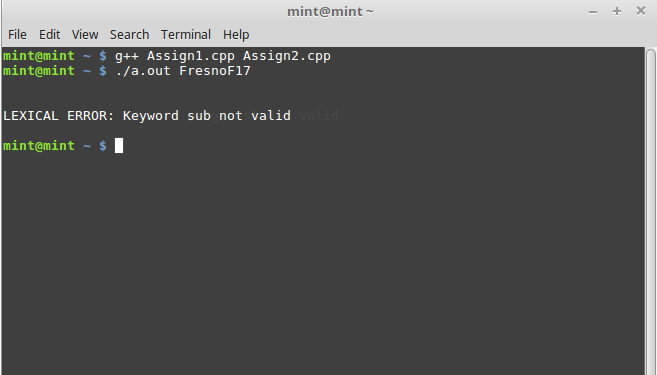
Output



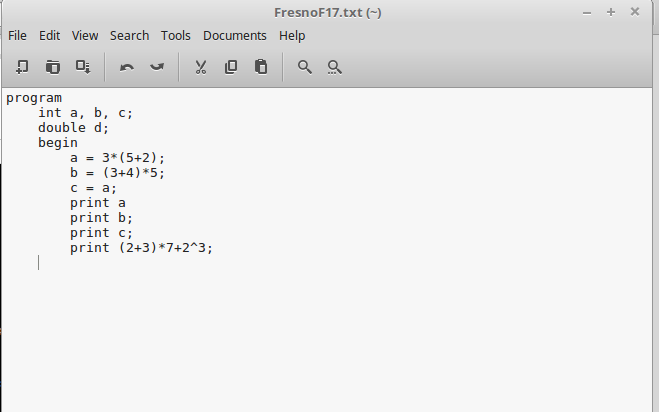
Input



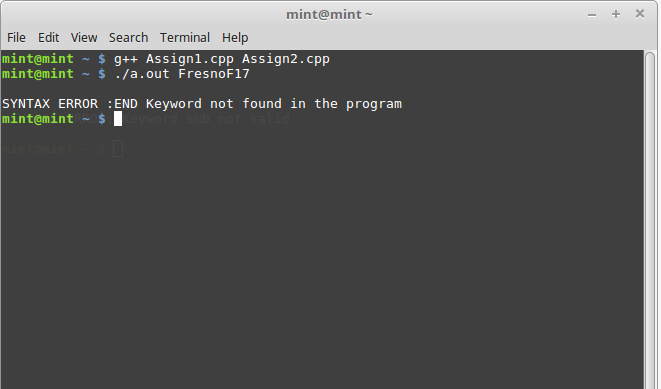
Output



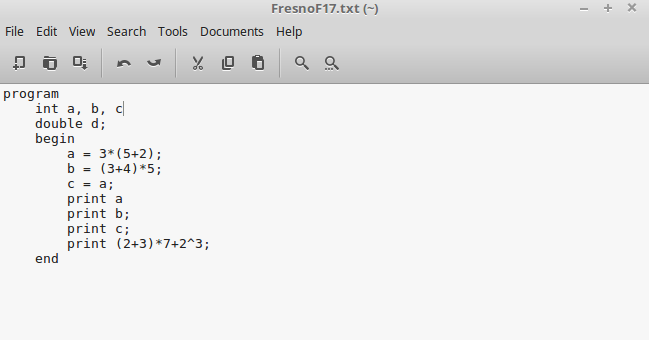
Input



Output



Input



Output

