Computer Science 323

Fall 2018

Final Project

Simon Lee

Cirjeffrey Baldomero

Method Used: Top-Down Parser (table I)

Language Used: C++

Original Grammer

|  |  |  |
| --- | --- | --- |
| <prog> | -> | **Program** <id> ; **var** <dec-list> **begin** <stat-list> **end** |
| <id> | -> | <letter> {<letter> | <digit>} |
| <dec-list> | -> | <dec> : <type> ; |
| <dec> | -> | <id> , <dec> | <id> |
| <type> | -> | **integer** |
| <stat-list> | -> | <stat>|<stat><stat-list> |
| <stat> | -> | <write> | <assign> |
| <write> | -> | **Show** (<id>); |
| <assign> | -> | <id> = <expr> ; |
| <expr> | -> | <expr> + <term> | <expr> - <term> | <term> |
| <term> | -> | <term> \* <factor> | <term> / <factor> | <factor> |
| <factor> | -> | <id> | <number> | (<expr>) |
| <number> | -> | <sign> <digit> {<digit>} |
| <sign> | -> | +|-|λ |
| <digit> | -> | 0|1|2|...|9 |
| <letter> | -> | a|b|c|d|e |

BNF Form

<prog> -> program <id>; var <dec-list>; begin <stat-list>; end

<id> -> <letter> <id2>

<id2> -> <letter> <id2>

<id2> -> <digit> <id2>

<id2> -> null

<dec-list> -> <dec>:<type>;

<dec> -> <id>, <dec2>

<dec2> -> <dec>

<dec2> -> <id>

<type> -> integer

<stat-list> -> <stat> <stat-list2>

<stat-list2> -> <stat>

<stat-list2> -> null

<stat> -> <write>

<stat> -> <assign>

<write> -> (<id>);

<assign> -> <id> = <expr>;

<expr> -> <expr> + <term>

<expr> -> <expr> - <term>

<expr> -> <term>

<term> -> <term> \* <factor>

<term> -> <term> / <factor>

<term> -> <factor>

<factor> -> <id>

<factor> -> <number>

<factor> -> (<expr>)

<number> -> <sign><digit><digit2>

<digit2> -> <digit> <digit2>

<digit2> -> null

<sign> -> +

<sign> -> -

<sign> -> null

<digit> -> 0

<digit> -> 1

<digit> -> 2

<digit> -> 3

<digit> -> 4

<digit> -> 5

<digit> -> 6

<digit> -> 7

<digit> -> 8

<digit> -> 9

<letter> -> a

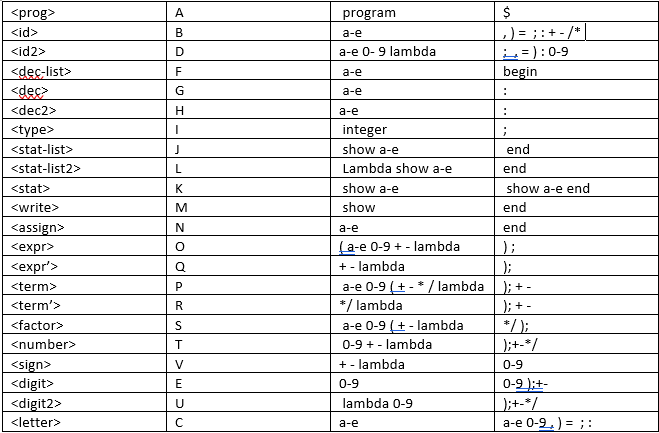
<letter> -> b

<letter> -> c

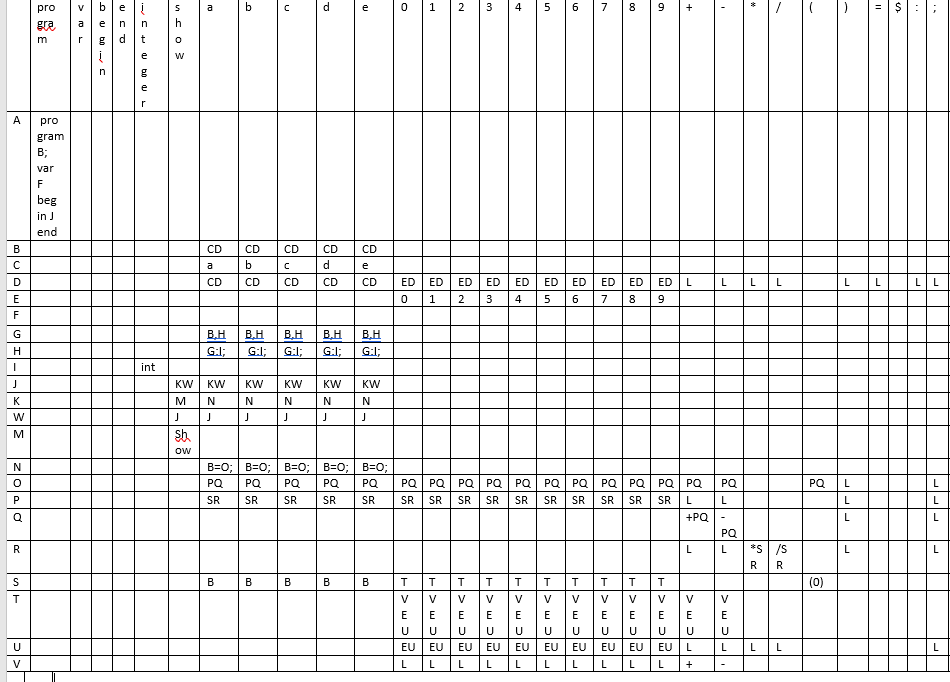
<letter> -> d

<letter> -> e

First And Follow



Parsing Table



Program For Part 1

//Program’s function is to take in a given text file and output it’s contents into another file with //proper syntax

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

using namespace std;

void writeFile(string);

bool comment = false;

bool singleComment = false;

void parseToken(string);

int main() {

string line;

ifstream inFile;

inFile.open("input.txt");

remove("output.txt"); // clears file during each run

while (getline(inFile, line)) { // while there are lines to get from file

istringstream iss(line); // convert line into separate tokens

if (line != "" && line[1] != '/') { // check if line is empty or single line comment only

while (iss) {

string tok;

iss >> tok; // converts tokens to strings to be analyzed

if (tok == "/\*") { // checks for multi line comment

comment = true;

}

if (tok== "\*/") {

comment = false;

break;

}

parseToken(tok);

}

}

}

system("pause");

}

// function parses each individual token.

void parseToken(string token) {

if (token[0] == '/' && token[1] == '/') { // checks for mid line comment

singleComment = true;

}

writeFile(token);

if (token == "" ) { // checks for end of line and pushes a newline

if (singleComment) { // checks for end of line for single line comments

singleComment = false;

}

writeFile("\n");

}

}

// opens and appends to file if not in comment state

void writeFile(string token) {

if (comment != true && singleComment != true) {

ofstream outFile;

outFile.open("output.txt", ::ios\_base::app);

outFile << " " << token;

outFile.close();

}

}

PROGRAM FOR PART 2

#include <iostream>

#include <stack>

#include <string>

#include <vector>

#include <sstream>

#include <iterator>

using namespace std;

void showStack(stack<string> s);

int getRow(string r);

int getCol(string c);

bool isAccepted(vector<string> w, size\_t length, stack<string>s) {

string table[22][32] = {

// P V B EN I S A B C D E 0 1 2 3 4 5 6 7 8 9 + - \* / ( ) = $ : ; ,

"program B ; var F begin J end","", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", //A

"", "", "", "", "", "", "C D", "C D", "C D", "C D", "C D", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", //B

"", "", "", "", "", "", "a", "b", "c", "d", "e", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//C

"", "", "", "", "", "", "C D", "C D", "C D", "C D", "C D", "E D", "E D", "E D", "E D", "E D", "E D", "E D", "E D", "E D", "E D", "L", "L", "L", "L", "", "L", "L", "", "L", "L", "L",//D

"", "", "", "", "", "", "", "", "", "", "", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "", "", "", "", "", "", "", "", "", "", "",//E

"", "", "", "", "", "", "G : I ;", "G : I ;", "G : I ;", "G : I ;", "G : I ;", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//F

"", "", "", "", "", "", "B , H", "B , H", "B , H", "B , H", "B , H", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//G

"", "", "", "", "", "", "G", "G", "G", "G", "G", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//H

"", "", "", "", "integer", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//I

"", "", "", "", "", "K W", "K W", "K W", "K W", "K W", "K W", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//J

"", "", "", "", "", "M", "N", "N", "N", "N", "N", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//K

"", "", "", "L", "", "J", "J", "J", "J", "J", "J", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//L

"", "", "", "", "", "show ( B ) ;", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//M

"", "", "", "", "", "", "B = O ;", "B = O ;", "B = O ;", "B = O ;", "B = O ;", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",//N

"", "", "", "", "", "", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "P Q", "", "", "P Q", "L", "", "", "", "L", "",//O

"", "", "", "", "", "", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "S R", "L", "L", "", "", "", "L", "", "", "", "L", "",//P

"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "+ P Q", "- P Q", "", "", "", "L", "", "", "", "L", "",//Q

"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "L", "L", "\* S R", "\* S R", "", "L", "", "", "", "L", "",//R

"", "", "", "", "", "", "B", "B", "B", "B", "B", "T", "T", "T", "T", "T", "T", "T", "T", "T", "T", "", "", "", "", "( O )", "", "", "", "", "", "",//S

"", "", "", "", "", "", "", "", "", "", "", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "V E U", "", "", "", "", "", "", "", "", "",

"", "", "", "", "", "", "", "", "", "", "", "E U", "E U", "E U", "E U", "E U", "E U", "E U", "E U", "E U", "E U", "L", "L", "L", "L", "", "", "", "", "", "L", "",//U

"", "", "", "", "", "", "", "", "", "", "", "L", "L", "L", "L", "L", "L", "L", "L", "L", "L", "+", "-", "", "", "", "", "", "", "", "", "", };//V

bool found = false;

string t; //the popped char

int r; //the row number

int c; // the column number

string e; //the string in specified index from table

vector<string> reserved\_words = { "program","var","begin","show","end","integer" };

//check for reserved words

for (int h = 0; h < reserved\_words.size(); h++) {

for (int g = 0; g < w.size(); g++) {

if (reserved\_words[h] == w[g]) {

found =true;

}

}

if (!found)

{

cout << reserved\_words[h] << " is expected" << endl;

return false;

}

}

string h;

bool is\_reserved;

s.push("$");

s.push("A");

t = s.top();

s.pop();

//this while loop will never finish because the function returns true or false

//once it matches $ or " " during each iteration

int num\_of\_commas = 0;

int curr\_comma = 0;

for (int k = 0; k < 1; k++) {

if (w[k] == ",")

num\_of\_commas++;

}

//check the program to determine if accepted or not

for (int j = 0; j < w.size(); j++) {

int i = 0; //read index for the terminal

int f; //the number of times we need to check if reserved word we only need to

//check once, otherwise check each individual character

int token\_size = 1; //for reserved words the size of token is the whole string

//for the other terminals its one

f = w[j].size(); //the amount of times we need to check

//check if the token is a reserved word

for (int z = 0; z < reserved\_words.size(); z++) {

if (w[j] == reserved\_words[z]) {

cout << "RESERVED WORD: " << w[j].substr(i, w[j].size()) << endl;

token\_size = w[j].size();

f = 1;

}

}

//check if variable is not defined

//check each token

while (i < f) {

int r = getRow(t);

int c = getCol(w[j].substr(i, token\_size));

e = table[r][c];

//error checking for ( is missing

if (t == "(" && w[j] == "a" || t == "(" && w[j] == "b" || t == "(" && w[j] == "c" ||

t == "(" && w[j] == "d" || t == "(" && w[j] == "e") {

cout << "( is missing " << endl;

return false;

}

//error checking for ) is missing

if (t == ")" && w[j] == ";") {

cout << ") is missing " << endl;

return false;

}

if (t == ",") { //keep count of the num of commas

curr\_comma++;

cout << "curr\_comma " << curr\_comma << endl;

}

if (t == "H") { //change dec to <id> in table if we reach the last variable declaration

if (curr\_comma == 3) {

cout << "change G to B" << endl;

e = "B";

}

}

if (t == "$") {

return true;

}

//match pop and go to next token in vector w

else if (t == w[j].substr(i, token\_size)) {

cout << "stack before popping after match" << endl;

showStack(s);

t = s.top();

s.pop();

i++;

}

else if (e == "") { //rejected

if (t == "D" && w[j] == "var" || t == "D" && w[j] == "begin" || t == "U" && w[j] != ";")

cout << ";" << " is missing" << endl;

else if (t == "R" && w[j] == "(" || t == "E" && w[j] == "+"|| t == "E" && w[j] == "-"|| t == "E" && w[j] == "\*"|| t == "E" && w[j] == "/")

cout << "illegal expression" << endl;

return false;

}

else if (e == "L") //if lambda pop stack

{

cout << "lambda at row " << r << "col " << c << endl;

t = s.top();

s.pop();

}

//push the entry in specified row and col

else {

cout << "pushing at row " << r << "col " << c << endl;

std::string str = e;

std::istringstream buf(str);

std::istream\_iterator<std::string> beg(buf), end;

std::vector<std::string> tokens(beg, end); // done!

for (auto& s : tokens)

std::cout << '"' << s << '"';

cout << endl;

for (int j = tokens.size() - 1; j >= 0; j--) {

s.push(tokens[j]);

}

cout << "stack after pushing e" << endl;

showStack(s);

t = s.top();

s.pop();

}

}

if (w[j] == "end") {

cout << "no error" << endl;

return t == "$";

}

}

return false;

}

void showStack(stack<string> s) {

for (std::stack<string> dump = s; !dump.empty(); dump.pop())

std::cout << dump.top() << " ";

}

int main() {

vector<string> arr = { "program" ,"a2018" , ";","var","ab1",",","cd",",","e33a",",","d18",":","integer",";","begin",

"ab1","=","33",";","cd","=","41",";","e33a","=","5",";","show", "(" , "ab1" , ")", ";" ,

"d18" , "=" , "ab1", "\*" , "(" , "cd" , "+" , "2" , "\*" , "e33a" , ")" , ";" , "show" , "(" ,

"d18" , ")" , ";" , "end"};

stack<string> s;

bool isaccep;

isaccep = isAccepted(arr, arr.size(), s);

if (isaccep)

cout << "accepted" << endl;

else

cout << "rejected" << endl;

system("pause");

return 0;

}

int getRow(string r) { //get row number associated with non-terminal

int x = 0;

if (r == "A") { x = 0; }

else if (r == "B") { x = 1; }

else if (r == "C") { x = 2; }

else if (r == "D") { x = 3; }

else if (r == "E") { x = 4; }

else if (r == "F") { x = 5; }

else if (r == "G") { x = 6; }

else if (r == "H") { x = 7; }

else if (r == "I") { x = 8; }

else if (r == "J") { x = 9; }

else if (r == "K") { x = 10; }

else if (r == "W") { x = 11; }

else if (r == "M") { x = 12; }

else if (r == "N") { x = 13; }

else if (r == "O") { x = 14; }

else if (r == "P") { x = 15; }

else if (r == "Q") { x = 16; }

else if (r == "R") { x = 17; }

else if (r == "S") { x = 18; }

else if (r == "T") { x = 19; }

else if (r == "U") { x = 20; }

else if (r == "V") { x = 21; }

return x;

}

int getCol(string c) { //get the column number associated with terminal

int y = 0;

if (c == "program") { y = 0; }

if (c == "var") { y = 1; }

if (c == "begin") { y = 2; }

if (c == "end") { y = 3; }

if (c == "integer") { y = 4; }

if (c == "show") { y = 5; }

if (c == "a") { y = 6; }

if (c == "b") { y = 7; }

if (c == "c") { y = 8; }

if (c == "d") { y = 9; }

if (c == "e") { y = 10; }

if (c == "0") { y = 11; }

if (c == "1") { y = 12; }

if (c == "2") { y = 13; }

if (c == "3") { y = 14; }

if (c == "4") { y = 15; }

if (c == "5") { y = 16; }

if (c == "6") { y = 17; }

if (c == "7") { y = 18; }

if (c == "8") { y = 19; }

if (c == "9") { y = 20; }

if (c == "+") { y = 21; }

if (c == "-") { y = 22; }

if (c == "\*") { y = 23; }

if (c == "/") { y = 24; }

if (c == "(") { y = 25; }

if (c == ")") { y = 26; }

if (c == "=") { y = 27; }

if (c == "$") { y = 28; }

if (c == ":") { y = 29; }

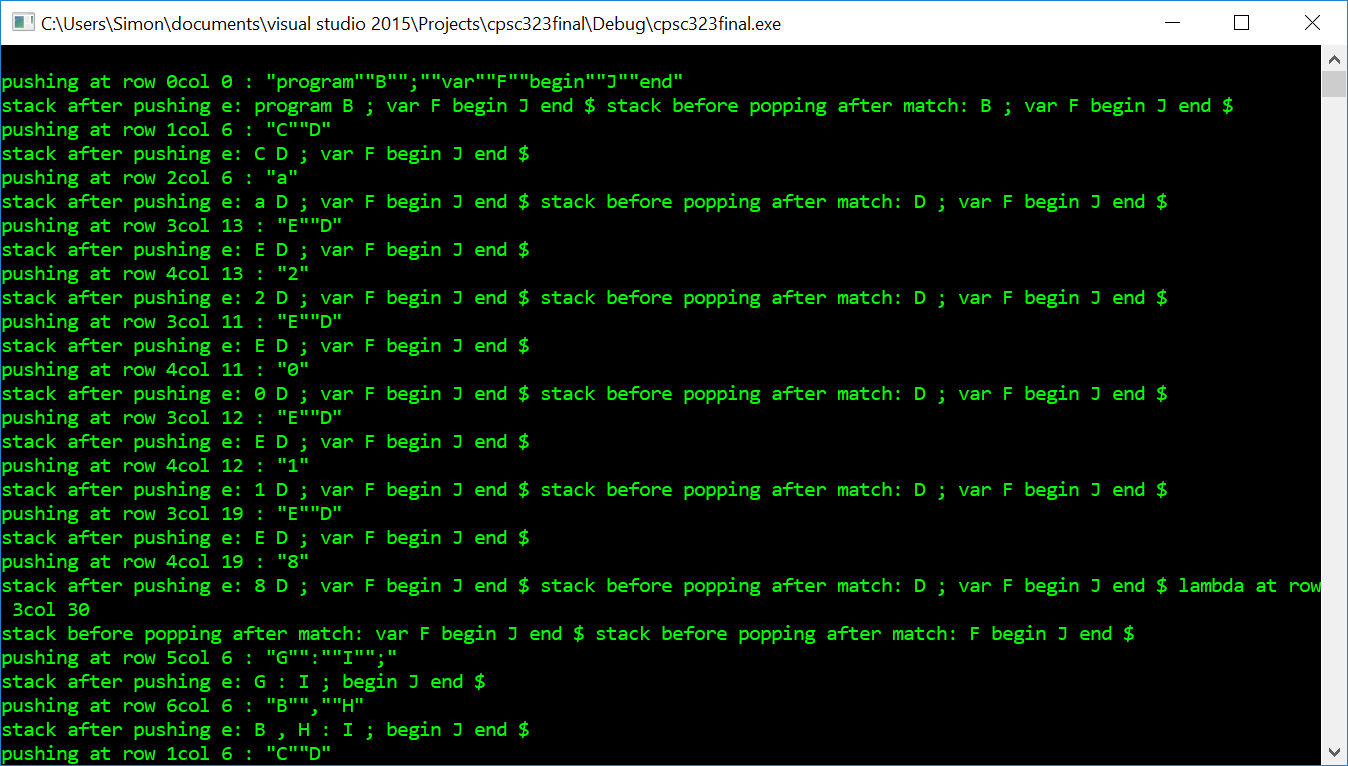
if (c == ";") { y = 30; }

if (c == ",") { y = 31; }

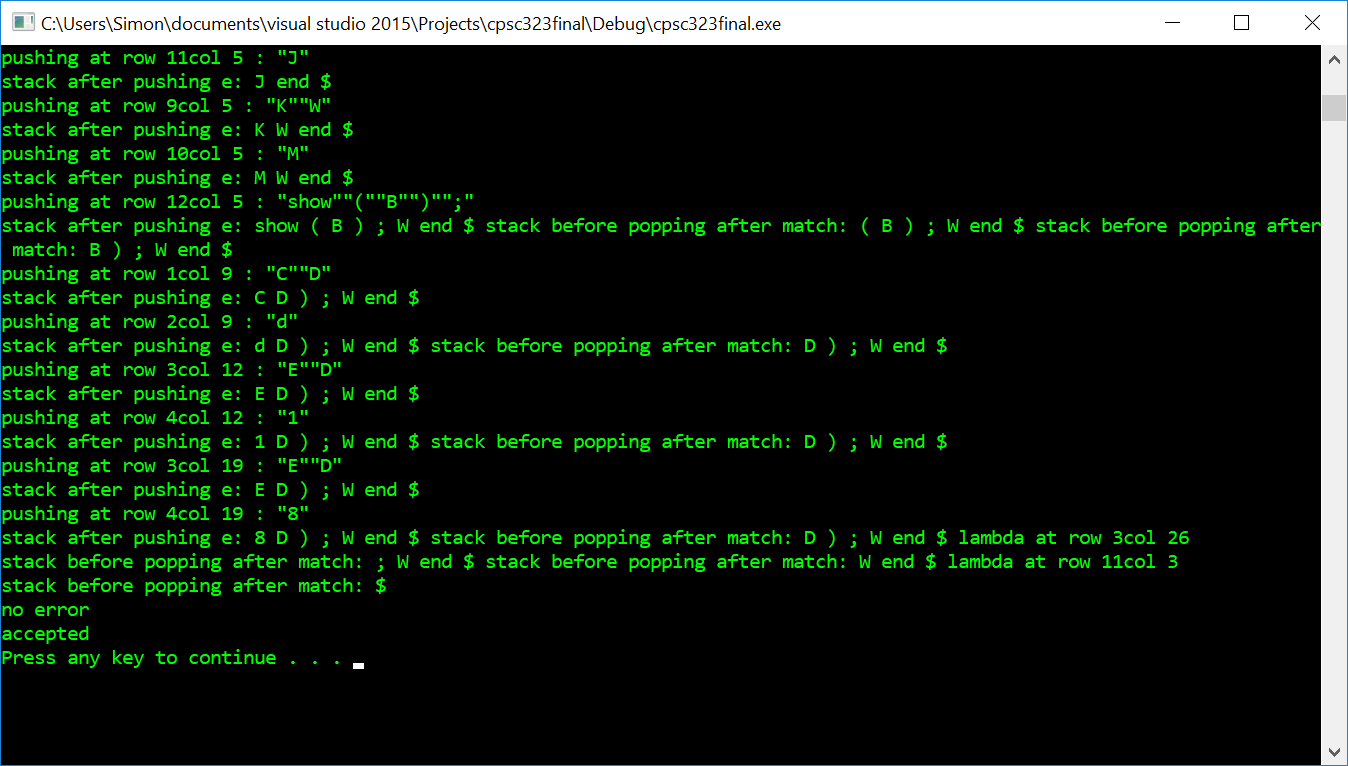
return y;

}

SAMPLE RUN



End result



PART 3 PROGRAM

#include <iostream>

#include <string>

using namespace std;

int ab1, cd, e33a, d18;

int main() {

ab1 = 33;

cd = 41;

e33a = 5;

cout << ab1<<endl;

d18 = ab1\*(cd + 2 \* e33a);

cout << d18 << endl;

system("pause");

return 0;

}

SAMPLE RUN

