Final Assessment: Compiler Lab

CSE-0302 Summer 2021

Md Al Yeasin, UG02-50-19-024
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
realalyeasin@gmail.com

***Abstract*—Syntax errors are very common in source programs. The main purpose of this session is to write programs to detect and report simple syntax errors.**

***Index Terms*—Syntax Errors, CFG, Parsing**

I. Introduction

Assignment 4 - Detecting Simple Syntax Errors- Syntax errors are very common in source programs. The main purpose of this session is to write programs to detect and report simple syntax errors. We performed detection of simple syntax errors like duplication of tokens except parentheses or braces, unbalanced braces or parentheses problem, unmatched 'else' problem.

Assignment 5.1 and 5.2 - Use of CFGs for Parsing- we implemented a simple recursive descent parser to parse a number of types of statements after exercising with simpler CFGs. We note that a recursive descent parser can be constructed from a CFG with reduced left recursion and ambiguity.

Assignment 6 - Predictive Parsing- we implemented LL(1) and LR(1) algorithms. Firstly, found the FIRST and FOLLOW sets of each of the non-terminals, then constructed the predictive parsing table for LL(1) method, demonstrated the moves of the LL(1) parser, LR(0) automaton for the grammar, parsing table for LR(1) parsing, then demonstrated the moves of the LR(1) parser

II. Literature Review

The term "lexical" in lexical analysis process of the compilation is derived from the word "lexeme", which is the basic conceptual unit of the linguistic morphological study. In computer science, lexical analysis, also referred to as lexing, scanning or tokenization, is the process of transforming the string of characters in source program to a stream of tokens, where the token is a string with a designated and identified meaning. It is the first phase of a two-step compilation processing model known as the analysis stage of compilation process used by compiler to understand the input source program. The objective is to convert character streams into words and recognize its token type. The generated stream of tokens is then used by the parser to determine the syntax of the source program. A program in compilation phase that performs a lexical analysis process is termed as lexical analyzer, lexer, scanner or tokenizer. Lexical analyzer is used in various computer science applications, such as word processing, information retrieval systems, pattern recognition systems and language-processing systems.However, the scope of our review study is related to language processing. Various tools are used for automatic generation of tokens and are more suitable for sequential execution of the process. Recent advances in multi-core architecture systems have led to the need to re-engineer the compilation process to integrate the multi-core architecture.

III. Proposed Methodology

Programming is basically solving a particular problem by giving coded instructions to the computer. Furthermore, the whole scenario of the programming cycle involves writing, testing, troubleshooting, debugging, and maintaining a computer program. Moreover, a good program should have clarity and simplicity of expressions, should make use of the proper name of identifiers, contain comments, and have a proper indentation. Besides, it should be free from all types of errors such as syntax errors, run time errors, and logical errors.

IV. Conclusion and Future Work

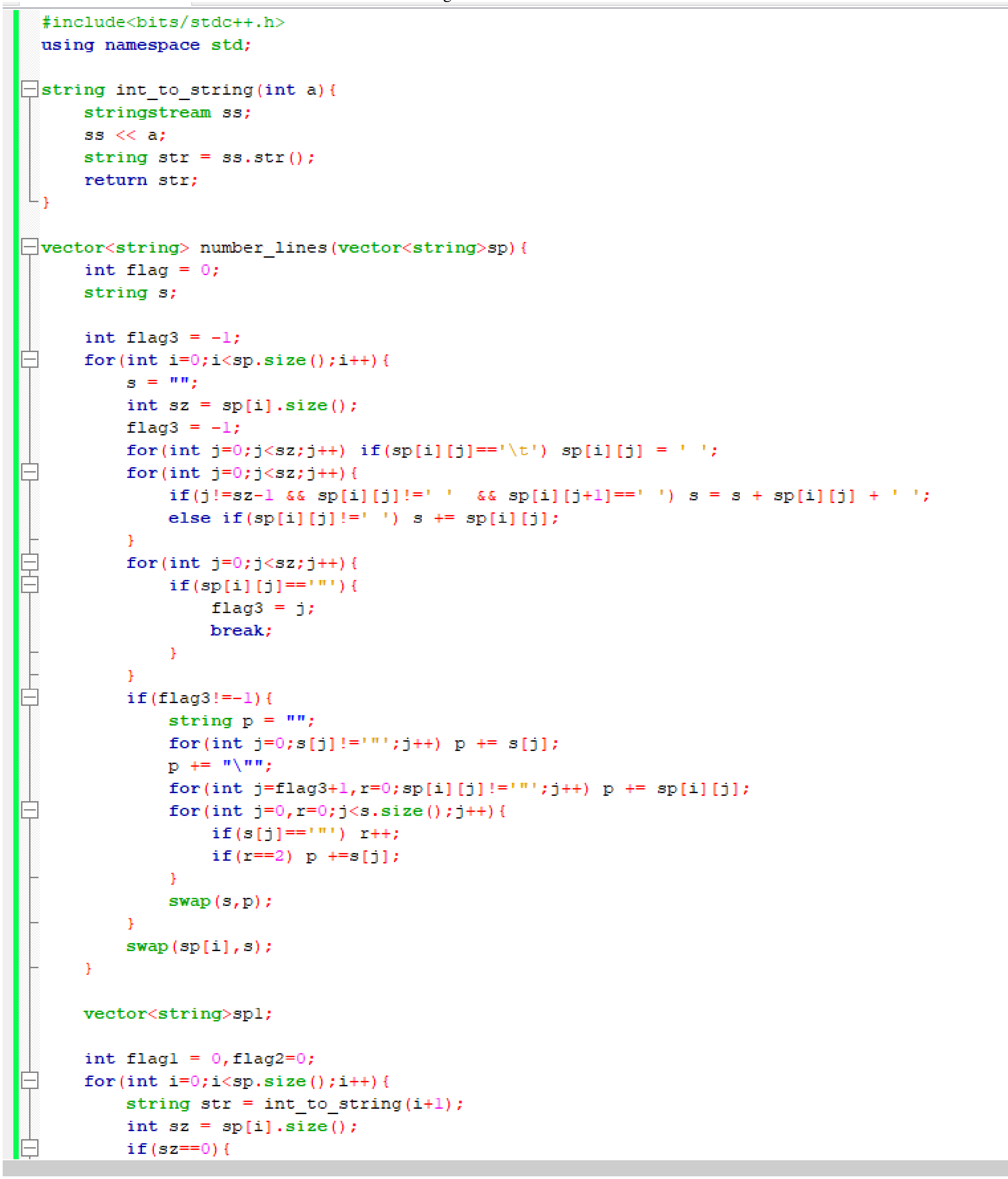
In future, we will develop this framework and try to make it more useful that will help making big projects in real life

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

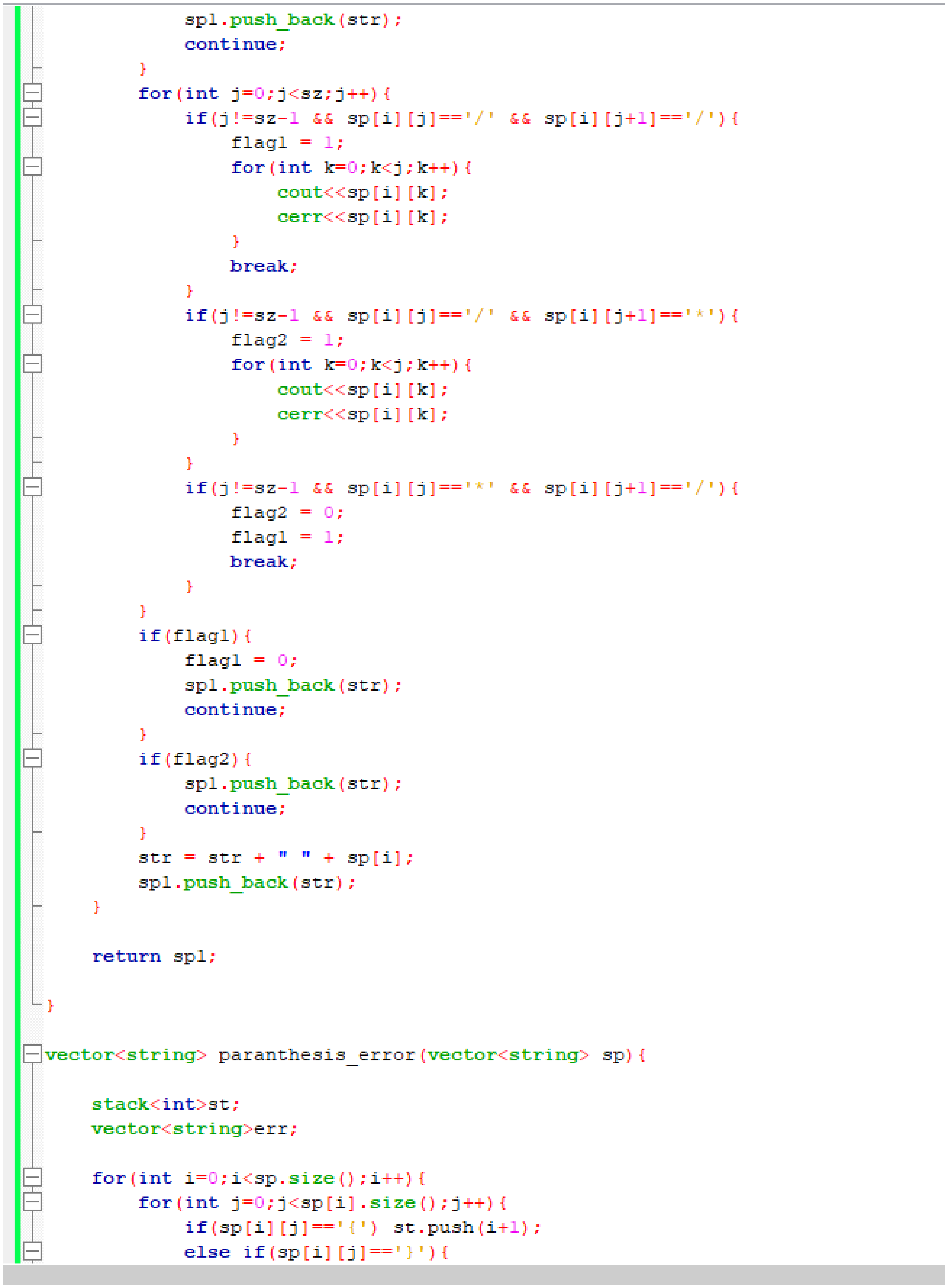
References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.



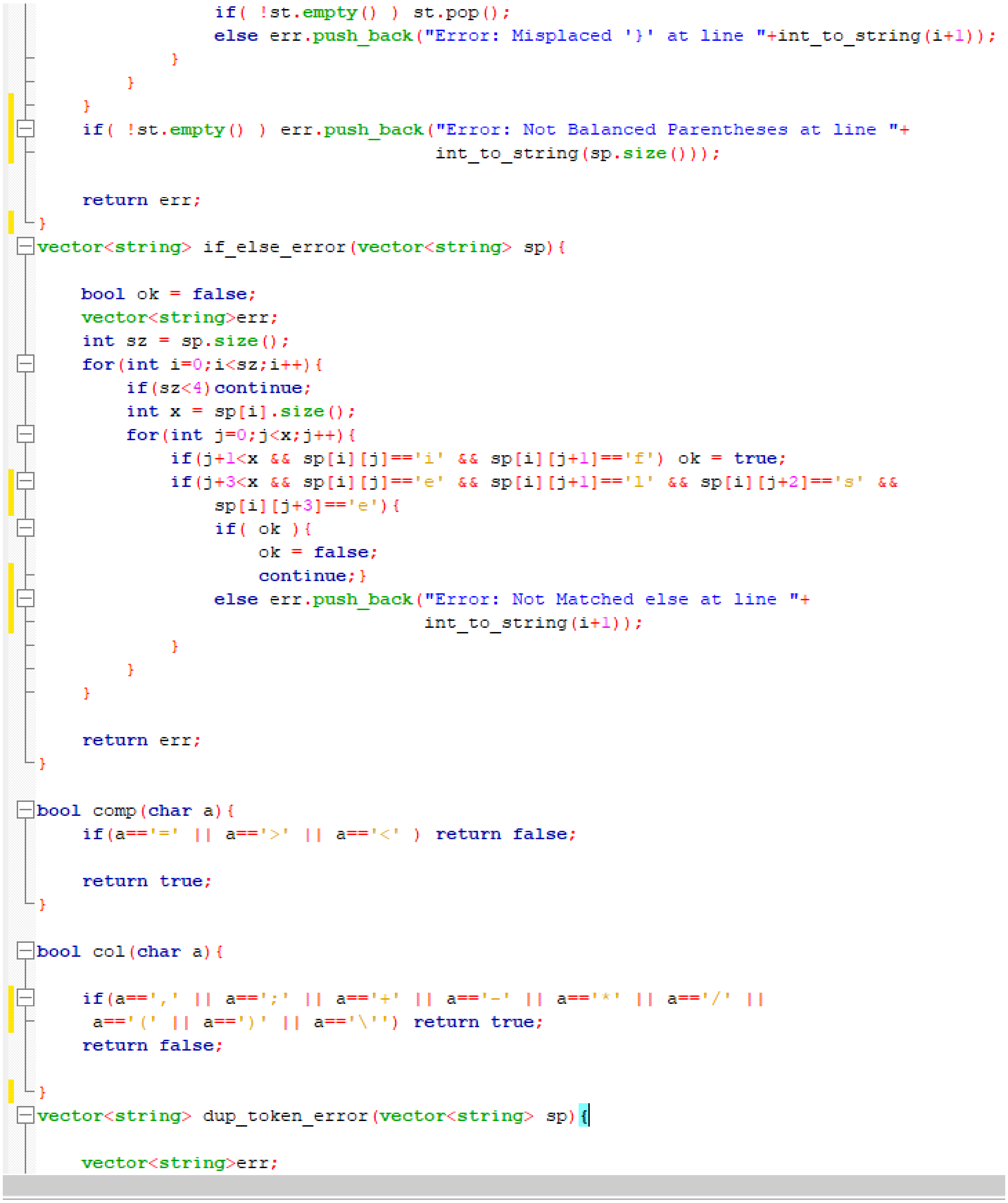
```
#include<bits/stdc++.h>
using namespace std;

string int_to_string(int a){
    stringstream ss;
    ss << a;
    string str = ss.str();
    return str;
}

vector<string> number_lines(vector<string>sp){
    int flag = 0;
    string s;

    int flag3 = -1;
    for(int i=0;i<sp.size();i++){
        s = "";
        int sz = sp[i].size();
        flag3 = -1;
        for(int j=0;j<sz;j++) if(sp[i][j]=='\t') sp[i][j] = ' ';
        for(int j=0;j<sz;j++){
            if(j!=sz-1 && sp[i][j]!=' '  && sp[i][j+1]==' ') s = s + sp[i][j] + ' ';
            else if(sp[i][j]!=' ') s += sp[i][j];
        }
        for(int j=0;j<sz;j++){
            if(sp[i][j]=='"'){
                flag3 = j;
                break;
            }
        }
        if(flag3!=-1){
            string p = "";
            for(int j=0;s[j]!='"';j++) p += s[j];
            p += "\"";
            for(int j=flag3+1,r=0;sp[i][j]!='"';j++) p += sp[i][j];
            for(int j=0,r=0;j<s.size();j++){
                if(s[j]=='"') r++;
                if(r==2) p +=s[j];
            }
            swap(s,p);
        }
        swap(sp[i],s);
    }

    vector<string>sp1;

    int flag1 = 0,flag2=0;
    for(int i=0;i<sp.size();i++){
        string str = int_to_string(i+1);
        int sz = sp[i].size();
        if(sz==0){
```

Fig. 1. Assignment 4 Page 1



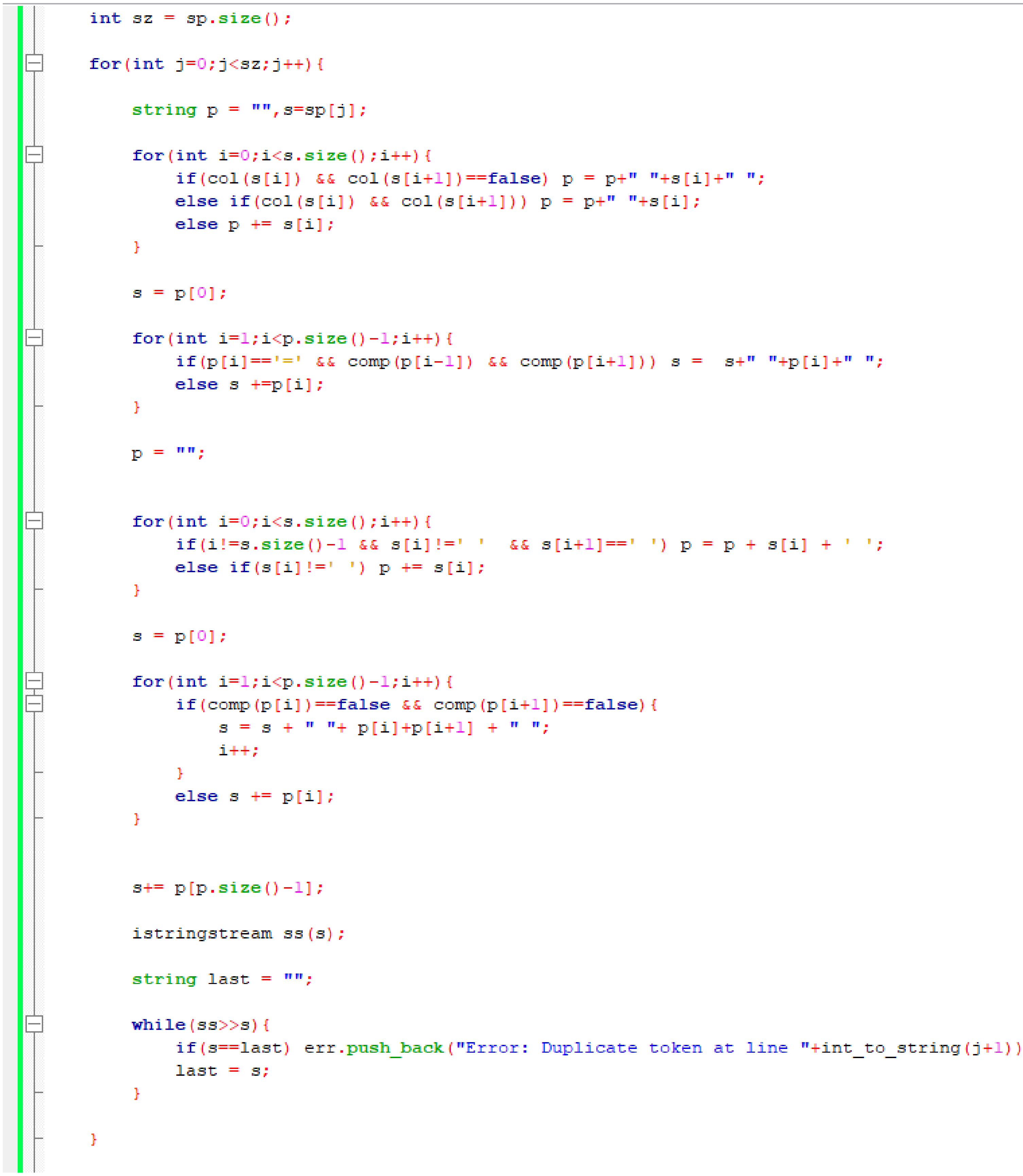
```
                spl.push_back(str);
                continue;
            }
            for(int j=0;j<sz;j++){
                if(j!=sz-1 && sp[i][j]=='/' && sp[i][j+1]=='/'){
                    flag1 = 1;
                    for(int k=0;k<j;k++){
                        cout<<sp[i][k];
                        cerr<<sp[i][k];
                    }
                    break;
                }
                if(j!=sz-1 && sp[i][j]=='/' && sp[i][j+1]=='*'){
                    flag2 = 1;
                    for(int k=0;k<j;k++){
                        cout<<sp[i][k];
                        cerr<<sp[i][k];
                    }
                }
                if(j!=sz-1 && sp[i][j]=='*' && sp[i][j+1]=='/'){
                    flag2 = 0;
                    flag1 = 1;
                    break;
                }
            }
            if(flag1){
                flag1 = 0;
                spl.push_back(str);
                continue;
            }
            if(flag2){
                spl.push_back(str);
                continue;
            }
            str = str + " " + sp[i];
            spl.push_back(str);
        }

        return spl;

}

vector<string> paranthesis_error(vector<string> sp){

    stack<int>st;
    vector<string>err;

    for(int i=0;i<sp.size();i++){
        for(int j=0;j<sp[i].size();j++){
            if(sp[i][j]=='{') st.push(i+1);
            else if(sp[i][j]=='}'){
```

Fig. 2. Assignment 4 Page 2



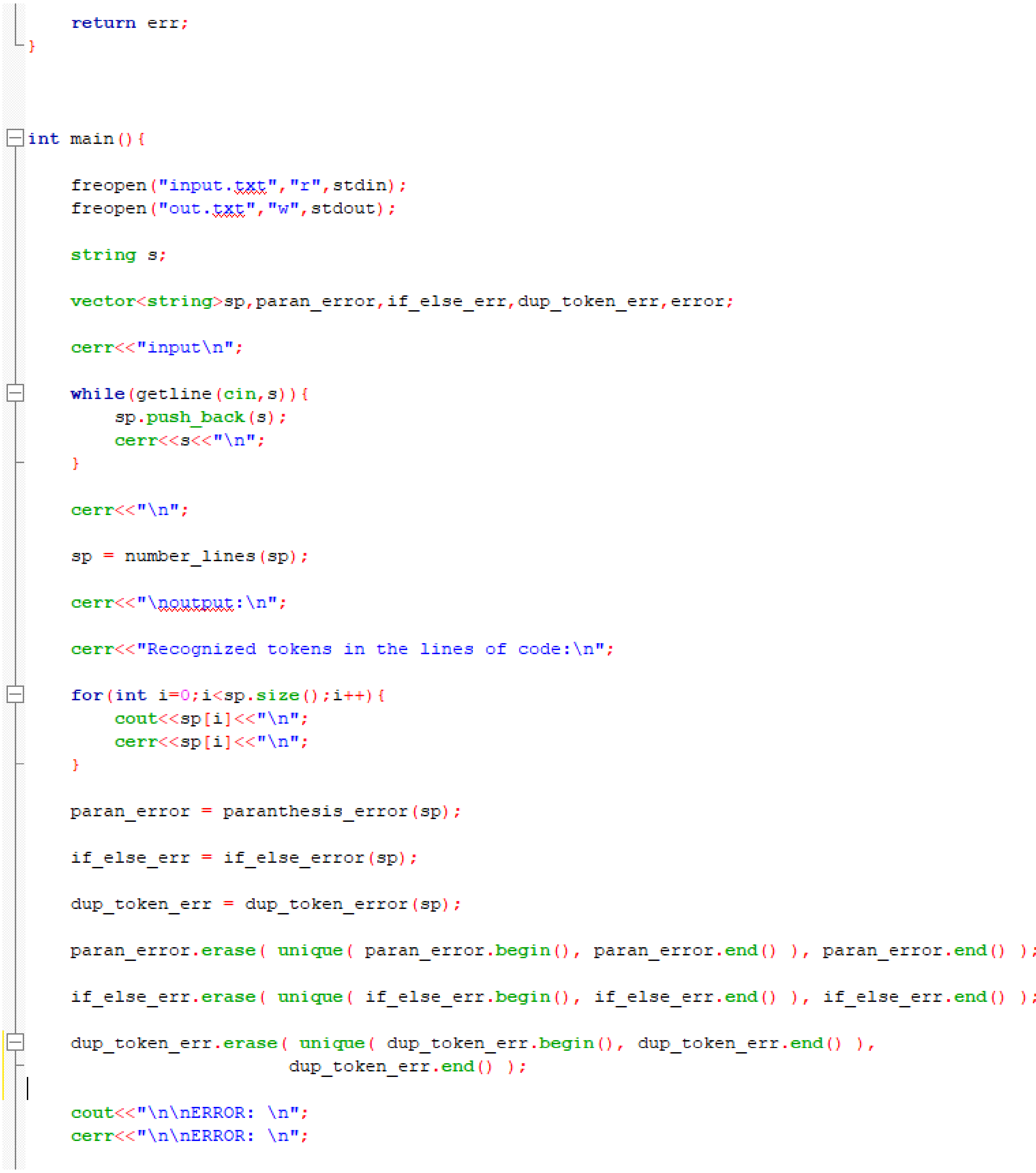
```
                    if( !st.empty() ) st.pop();
                    else err.push_back("Error: Misplaced '}' at line "+int_to_string(i+1));
                }
            }
        }
        if( !st.empty() ) err.push_back("Error: Not Balanced Parentheses at line "+
                                        int_to_string(sp.size()));

        return err;
}
vector<string> if_else_error(vector<string> sp){

        bool ok = false;
        vector<string>err;
        int sz = sp.size();
        for(int i=0;i<sz;i++){
            if(sz<4)continue;
            int x = sp[i].size();
            for(int j=0;j<x;j++){
                if(j+1<x && sp[i][j]=='i' && sp[i][j+1]=='f') ok = true;
                if(j+3<x && sp[i][j]=='e' && sp[i][j+1]=='l' && sp[i][j+2]=='s' &&
                    sp[i][j+3]=='e'){
                    if( ok ){
                        ok = false;
                        continue;}
                    else err.push_back("Error: Not Matched else at line "+
                                       int_to_string(i+1));
                }
            }
        }

        return err;
}

bool comp(char a){
        if(a=='=' || a=='>' || a=='<' ) return false;

        return true;
}

bool col(char a){

        if(a==',' || a==';' || a=='+' || a=='-' || a=='*' || a=='/' ||
         a=='(' || a==')' || a=='\'') return true;
        return false;

}
vector<string> dup_token_error(vector<string> sp){

        vector<string>err;
```

Fig. 3. Assignment 4 Page 3



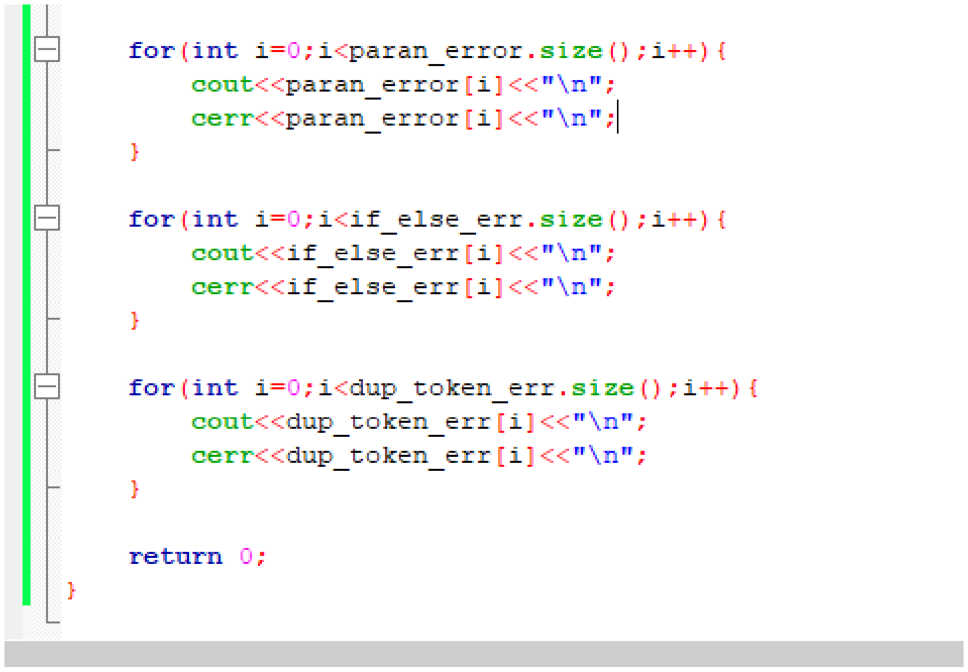
```
int sz = sp.size();

for(int j=0;j<sz;j++){

    string p = "",s=sp[j];

    for(int i=0;i<s.size();i++){
        if(col(s[i]) && col(s[i+1])==false) p = p+" "+s[i]+" ";
        else if(col(s[i]) && col(s[i+1])) p = p+" "+s[i];
        else p += s[i];
    }

    s = p[0];

    for(int i=1;i<p.size()-1;i++){
        if(p[i]=='=' && comp(p[i-1]) && comp(p[i+1])) s =  s+" "+p[i]+" ";
        else s +=p[i];
    }

    p = "";


    for(int i=0;i<s.size();i++){
        if(i!=s.size()-1 && s[i]!=' '  && s[i+1]==' ') p = p + s[i] + ' ';
        else if(s[i]!=' ') p += s[i];
    }

    s = p[0];

    for(int i=1;i<p.size()-1;i++){
        if(comp(p[i])==false && comp(p[i+1])==false){
            s = s + " "+ p[i]+p[i+1] + " ";
            i++;
        }
        else s += p[i];
    }


    s+= p[p.size()-1];

    istringstream ss(s);

    string last = "";

    while(ss>>s){
        if(s==last) err.push_back("Error: Duplicate token at line "+int_to_string(j+1))
        last = s;
    }

}
```

Fig. 4.   Assignment 4 Page 4



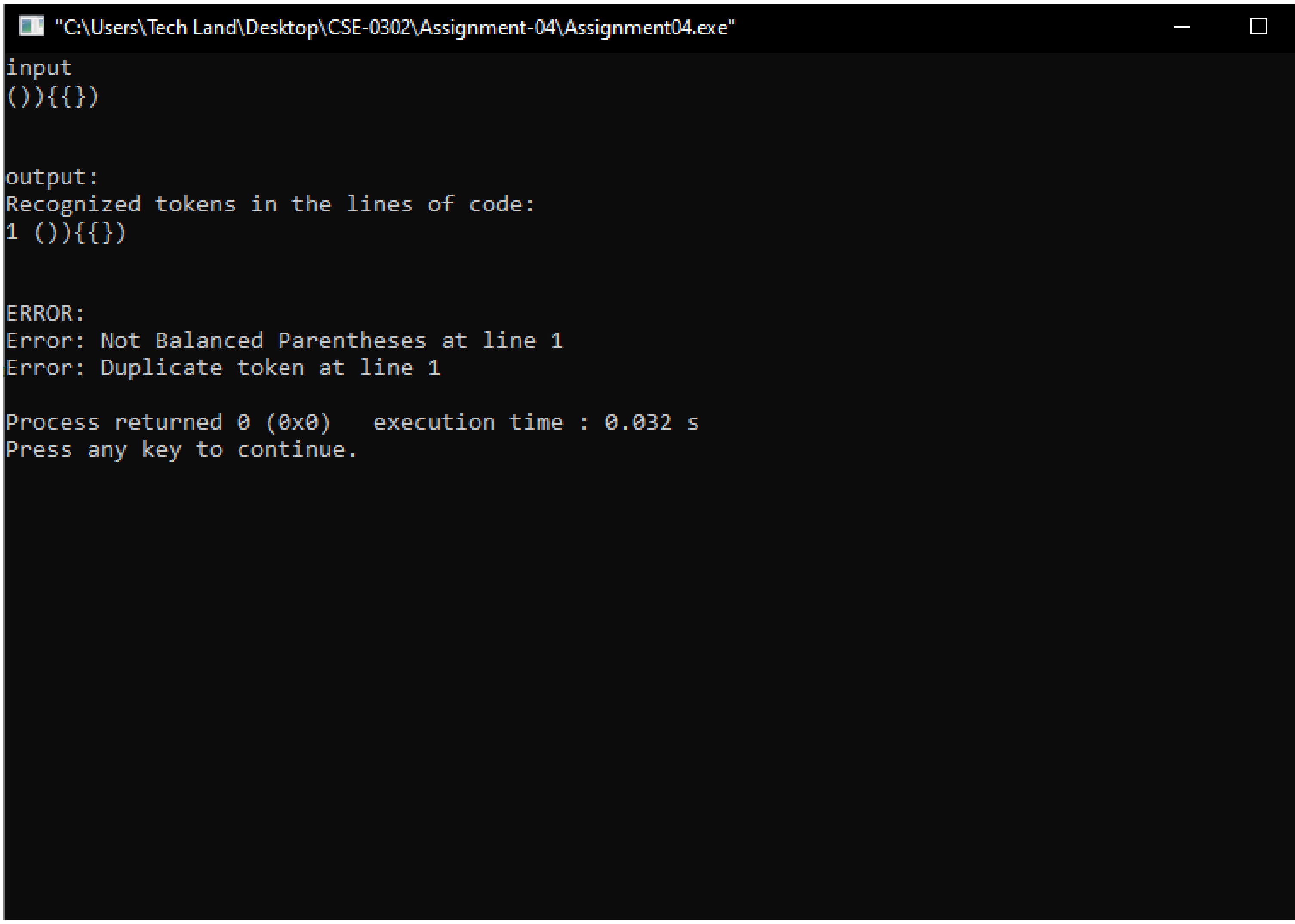
```
    return err;
}



int main(){

    freopen("input.txt","r",stdin);
    freopen("out.txt","w",stdout);

    string s;

    vector<string>sp,paran_error,if_else_err,dup_token_err,error;

    cerr<<"input\n";

    while(getline(cin,s)){
        sp.push_back(s);
        cerr<<s<<"\n";
    }

    cerr<<"\n";

    sp = number_lines(sp);

    cerr<<"\noutput:\n";

    cerr<<"Recognized tokens in the lines of code:\n";

    for(int i=0;i<sp.size();i++){
        cout<<sp[i]<<"\n";
        cerr<<sp[i]<<"\n";
    }

    paran_error = paranthesis_error(sp);

    if_else_err = if_else_error(sp);

    dup_token_err = dup_token_error(sp);

    paran_error.erase( unique( paran_error.begin(), paran_error.end() ), paran_error.end() );

    if_else_err.erase( unique( if_else_err.begin(), if_else_err.end() ), if_else_err.end() );

    dup_token_err.erase( unique( dup_token_err.begin(), dup_token_err.end() ),
                        dup_token_err.end() );

    cout<<"\n\nERROR: \n";
    cerr<<"\n\nERROR: \n";
```

Fig. 5. Assignment 4 Page 5



```
    for(int i=0;i<paran_error.size();i++){
        cout<<paran_error[i]<<"\n";
        cerr<<paran_error[i]<<"\n";
    }

    for(int i=0;i<if_else_err.size();i++){
        cout<<if_else_err[i]<<"\n";
        cerr<<if_else_err[i]<<"\n";
    }

    for(int i=0;i<dup_token_err.size();i++){
        cout<<dup_token_err[i]<<"\n";
        cerr<<dup_token_err[i]<<"\n";
    }

    return 0;
}
```

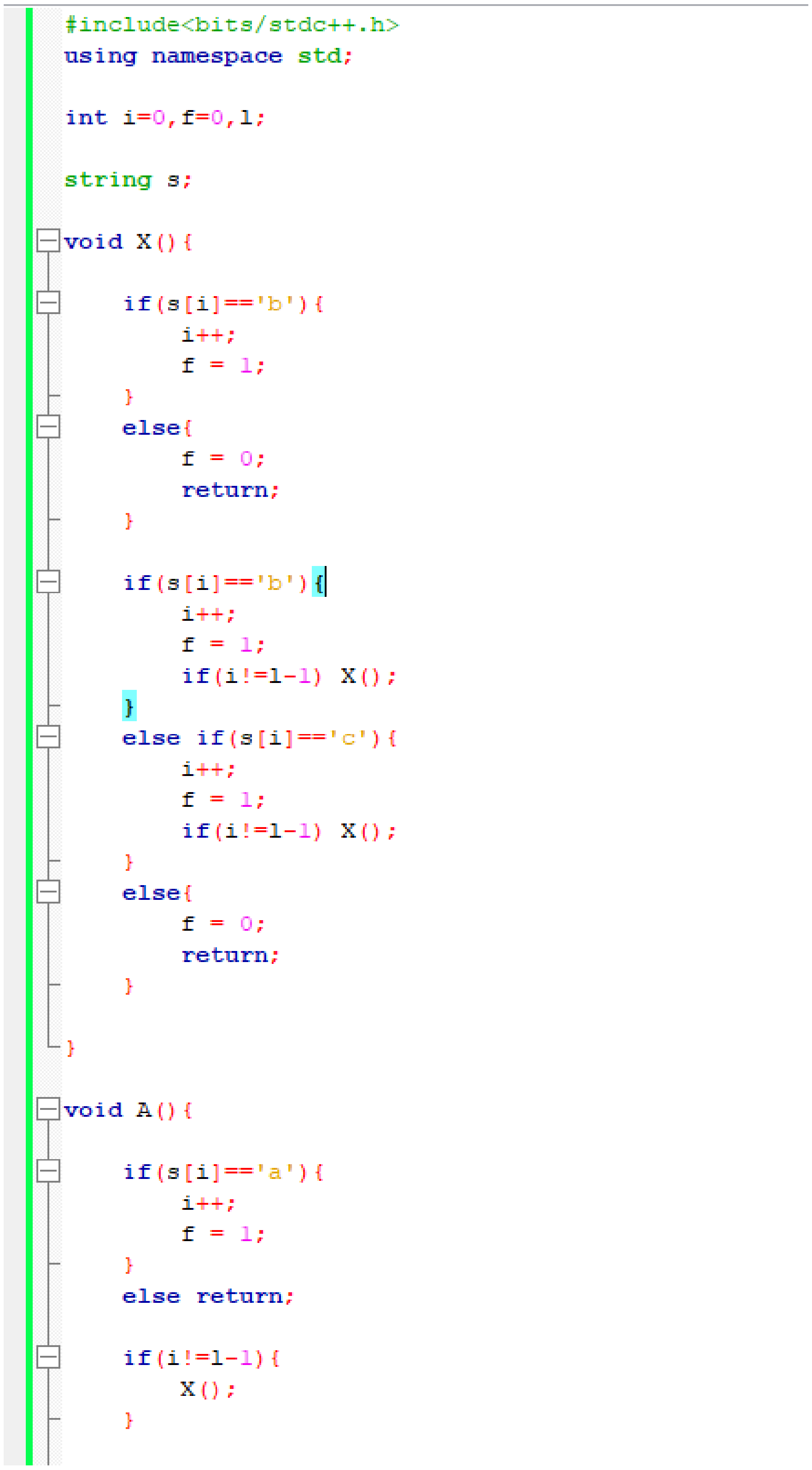
Fig. 6. Assignment 4 Page 6


"C:\Users\Tech Land\Desktop\CSE-0302\Assignment-04\Assignment04.exe"

```
input
()){{})


output:
Recognized tokens in the lines of code:
1 ()){{})


ERROR:
Error: Not Balanced Parentheses at line 1
Error: Duplicate token at line 1

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

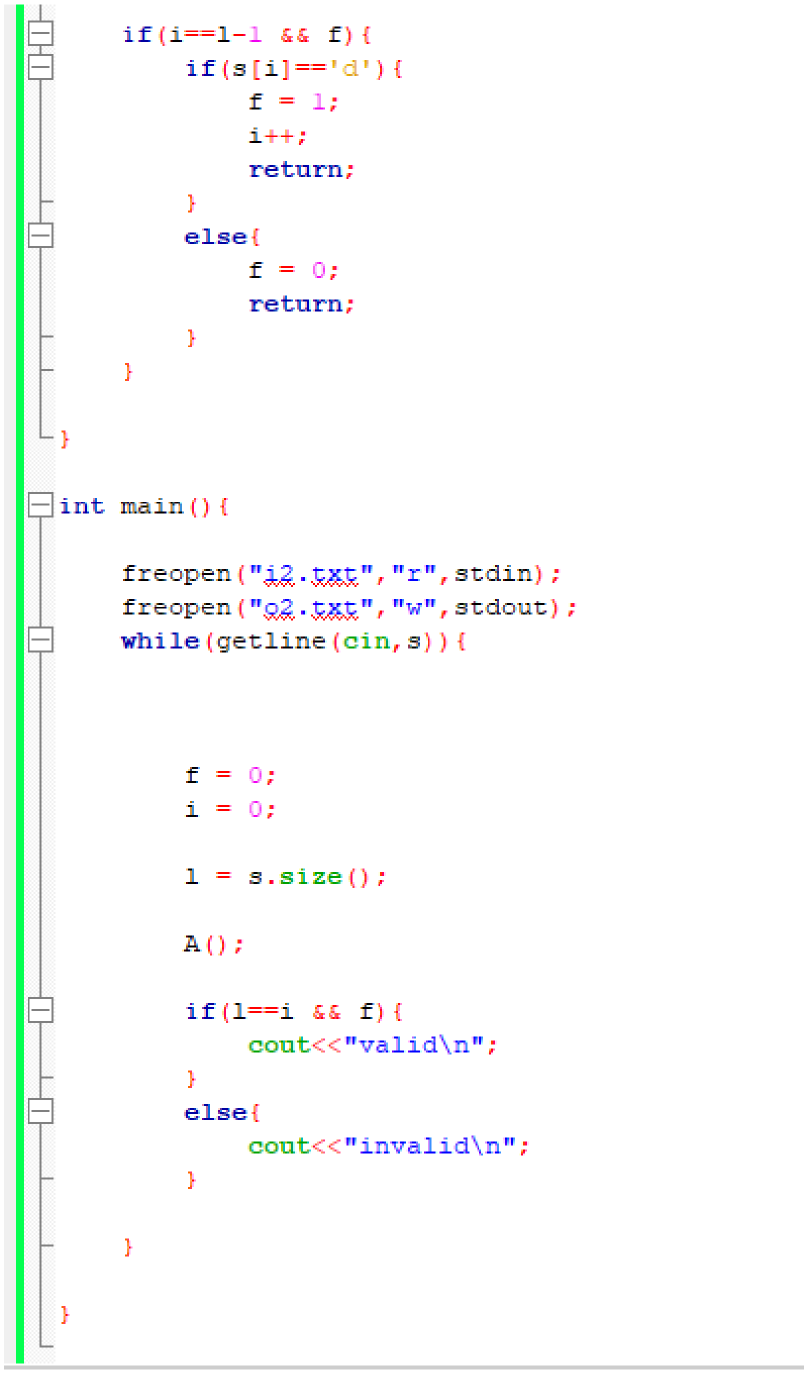
Fig. 7. Assignment 4 Page 7

Assignment 5.1



```
#include<bits/stdc++.h>
using namespace std;

int i=0,f=0,l;

string s;

void X(){

    if(s[i]=='b'){
        i++;
        f = 1;
    }
    else{
        f = 0;
        return;
    }

    if(s[i]=='b'){
        i++;
        f = 1;
        if(i!=l-1) X();
    }
    else if(s[i]=='c'){
        i++;
        f = 1;
        if(i!=l-1) X();
    }
    else{
        f = 0;
        return;
    }

}

void A(){

    if(s[i]=='a'){
        i++;
        f = 1;
    }
    else return;

    if(i!=l-1){
        X();
    }
```

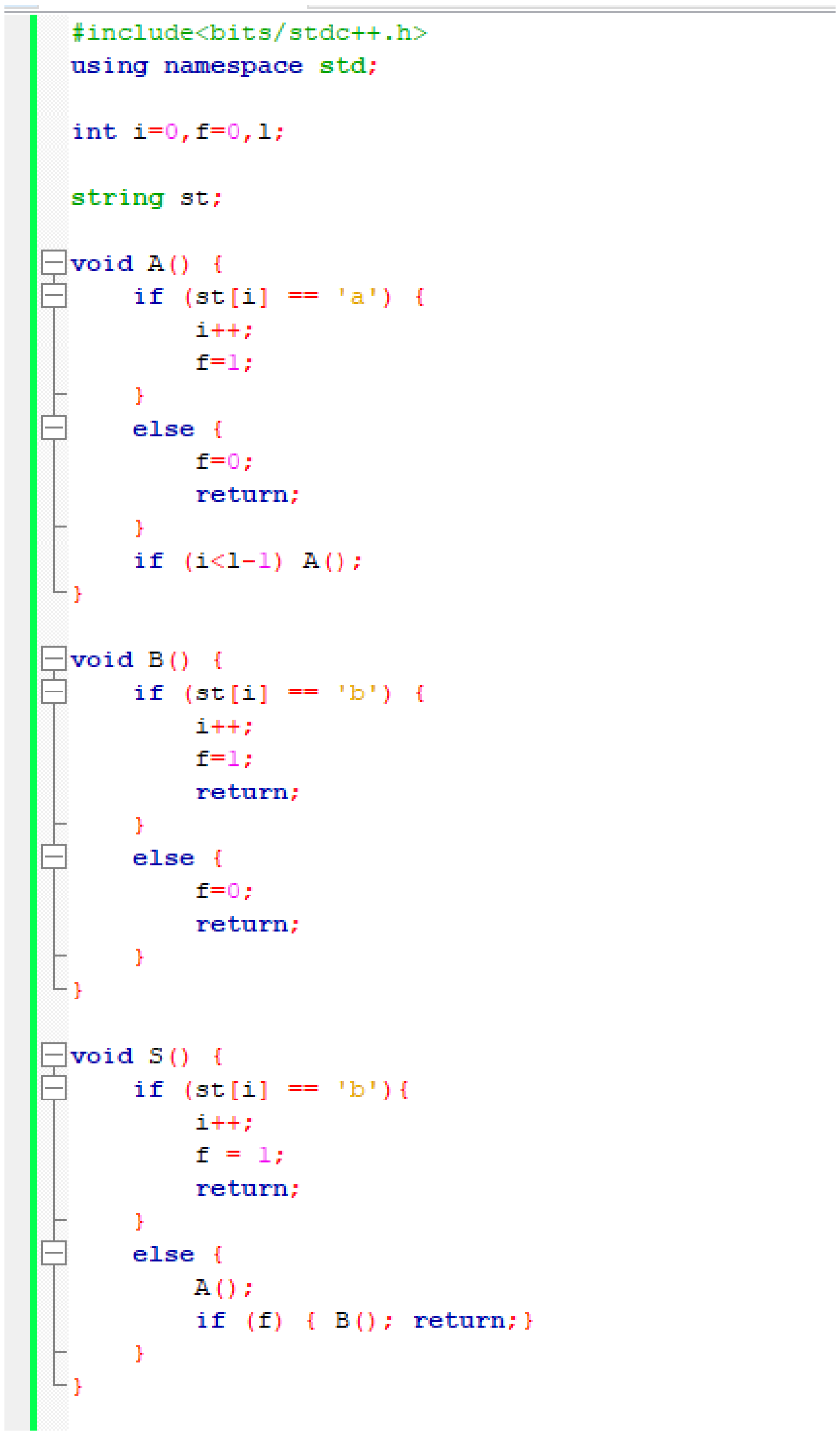
Fig. 8. Assignment 5.1 Page 1



```
    if(i==l-1 && f){
        if(s[i]=='d'){
            f = 1;
            i++;
            return;
        }
        else{
            f = 0;
            return;
        }
    }

}

int main(){

    freopen("i2.txt","r",stdin);
    freopen("o2.txt","w",stdout);
    while(getline(cin,s)){



        f = 0;
        i = 0;

        l = s.size();

        A();

        if(l==i && f){
            cout<<"valid\n";
        }
        else{
            cout<<"invalid\n";
        }

    }

}
```

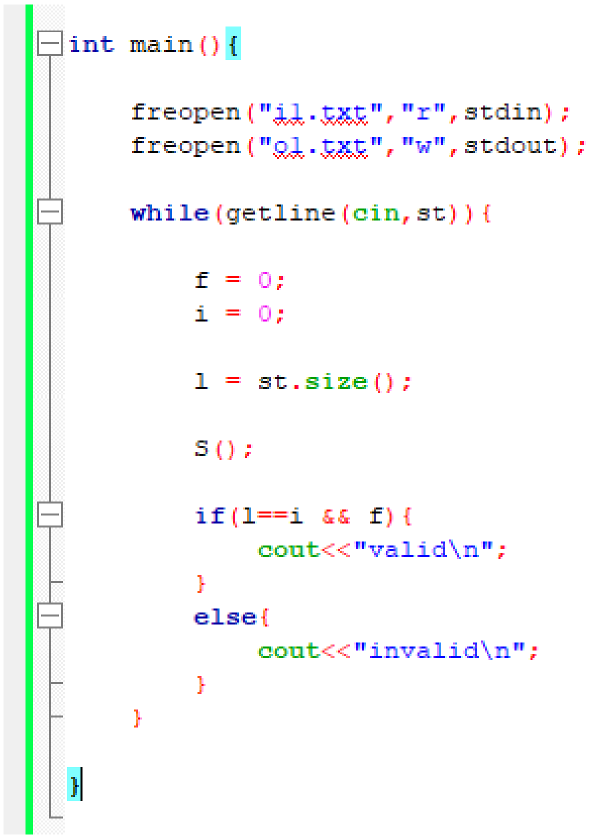
Fig. 9. Assignment 5.1 Page 2

Assignment 5.2



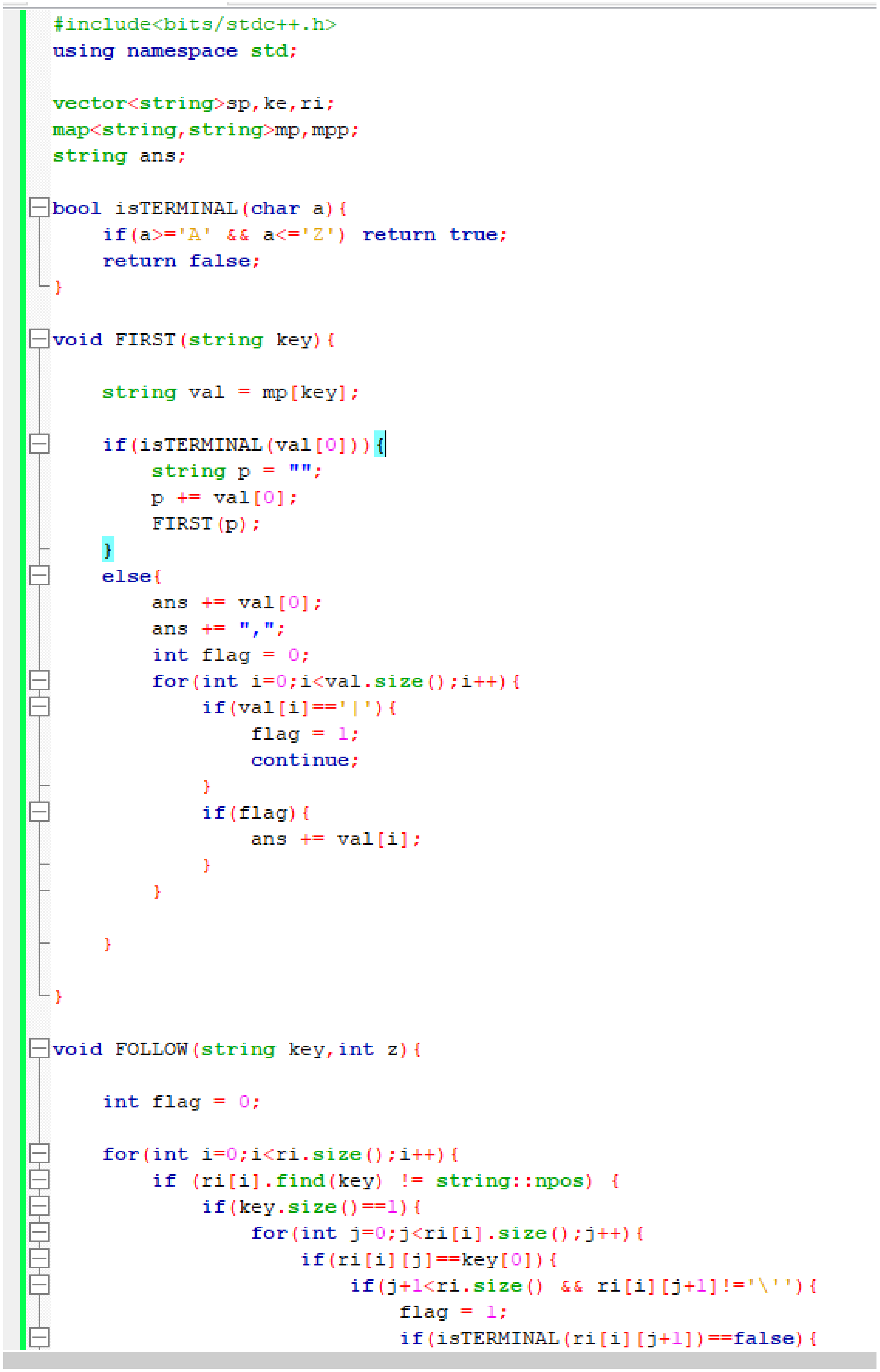
```
#include<bits/stdc++.h>
using namespace std;

int i=0,f=0,l;

string st;

void A() {
    if (st[i] == 'a') {
        i++;
        f=1;
    }
    else {
        f=0;
        return;
    }
    if (i<l-1) A();
}

void B() {
    if (st[i] == 'b') {
        i++;
        f=1;
        return;
    }
    else {
        f=0;
        return;
    }
}

void S() {
    if (st[i] == 'b'){
        i++;
        f = 1;
        return;
    }
    else {
        A();
        if (f) { B(); return;}
    }
}
```

Fig. 10. Assignment 5.2 Page 1



```
int main(){

    freopen("i1.txt","r",stdin);
    freopen("o1.txt","w",stdout);

    while(getline(cin,st)){

        f = 0;
        i = 0;

        l = st.size();

        S();

        if(l==i && f){
            cout<<"valid\n";
        }
        else{
            cout<<"invalid\n";
        }
    }

}
```

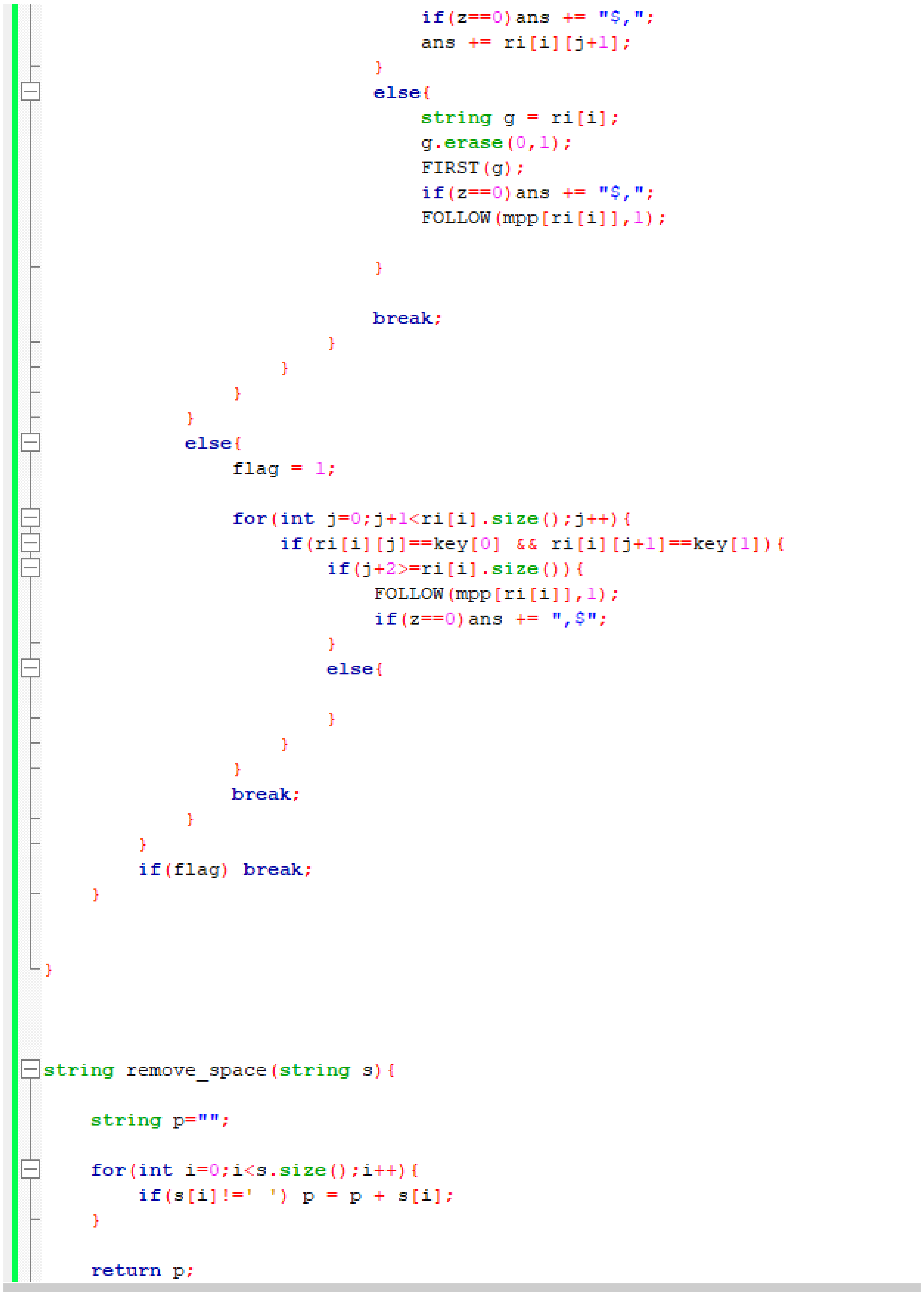
Fig. 11. Assignment 5.2 Page 2



```
#include<bits/stdc++.h>
using namespace std;

vector<string>sp,ke,ri;
map<string,string>mp,mpp;
string ans;

bool isTERMINAL(char a){
    if(a>='A' && a<='Z') return true;
    return false;
}

void FIRST(string key){

    string val = mp[key];

    if(isTERMINAL(val[0])){
        string p = "";
        p += val[0];
        FIRST(p);
    }
    else{
        ans += val[0];
        ans += ",";
        int flag = 0;
        for(int i=0;i<val.size();i++){
            if(val[i]=='|'){
                flag = 1;
                continue;
            }
            if(flag){
                ans += val[i];
            }
        }

    }

}

void FOLLOW(string key,int z){

    int flag = 0;

    for(int i=0;i<ri.size();i++){
        if (ri[i].find(key) != string::npos) {
            if(key.size()==1){
                for(int j=0;j<ri[i].size();j++){
                    if(ri[i][j]==key[0]){
                        if(j+1<ri.size() && ri[i][j+1]!='\''){
                            flag = 1;
                            if(isTERMINAL(ri[i][j+1])==false){
```

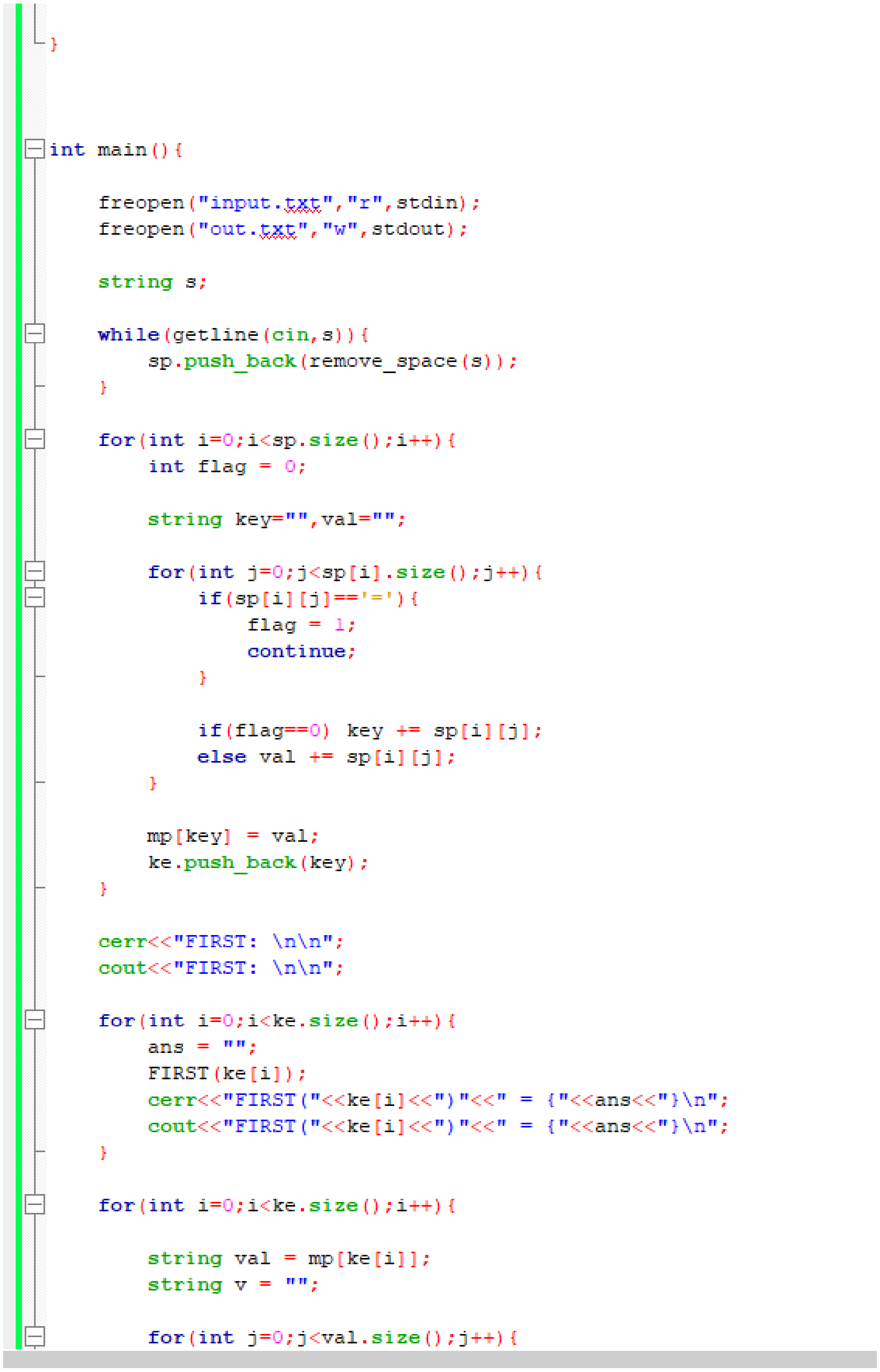
Assignment 6

Fig. 12. Assignment 6 Page 1



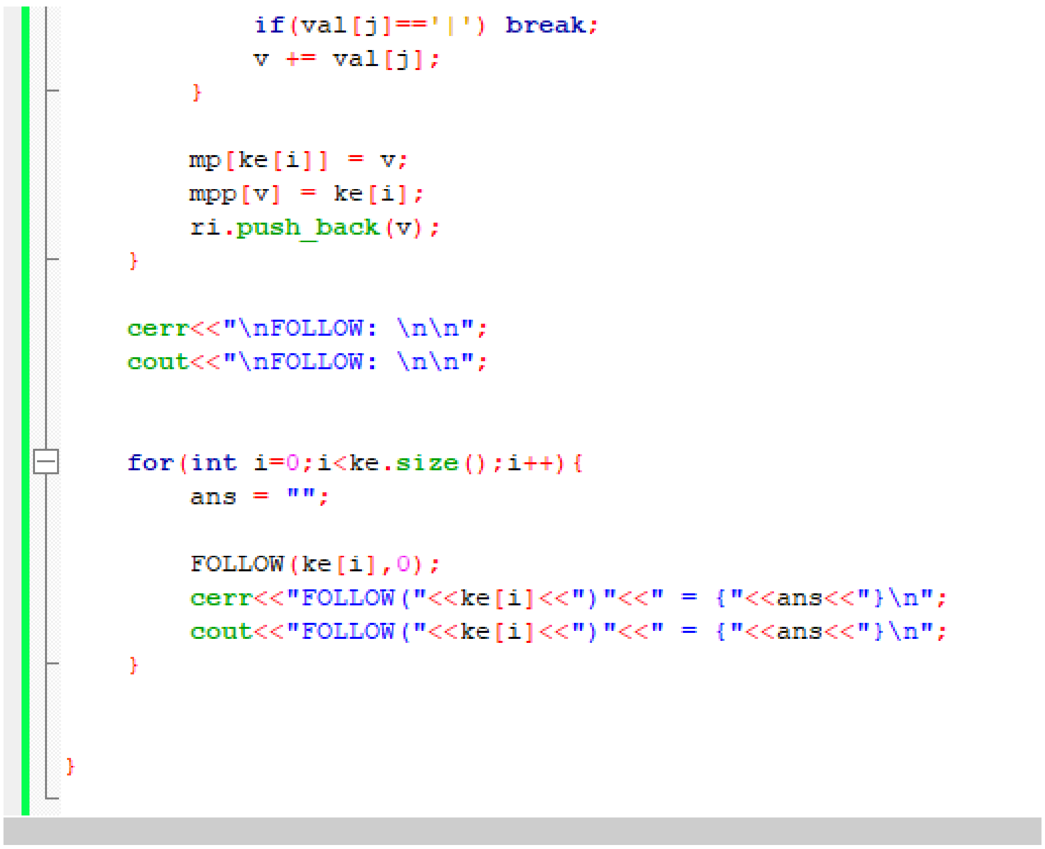
```
                                if(z==0)ans += "$,";
                                ans += ri[i][j+1];
                            }
                            else{
                                string g = ri[i];
                                g.erase(0,1);
                                FIRST(g);
                                if(z==0)ans += "$,";
                                FOLLOW(mpp[ri[i]],1);

                            }

                            break;
                        }
                    }
                }
            }
            else{
                flag = 1;

                for(int j=0;j+1<ri[i].size();j++){
                    if(ri[i][j]==key[0] && ri[i][j+1]==key[1]){
                        if(j+2>=ri[i].size()){
                            FOLLOW(mpp[ri[i]],1);
                            if(z==0)ans += ",$";
                        }
                        else{

                        }
                    }
                }
                break;
            }
        }
        if(flag) break;
    }


}



string remove_space(string s){

    string p="";

    for(int i=0;i<s.size();i++){
        if(s[i]!=' ') p = p + s[i];
    }

    return p;
```

Fig. 13. Assignment 6 Page 2



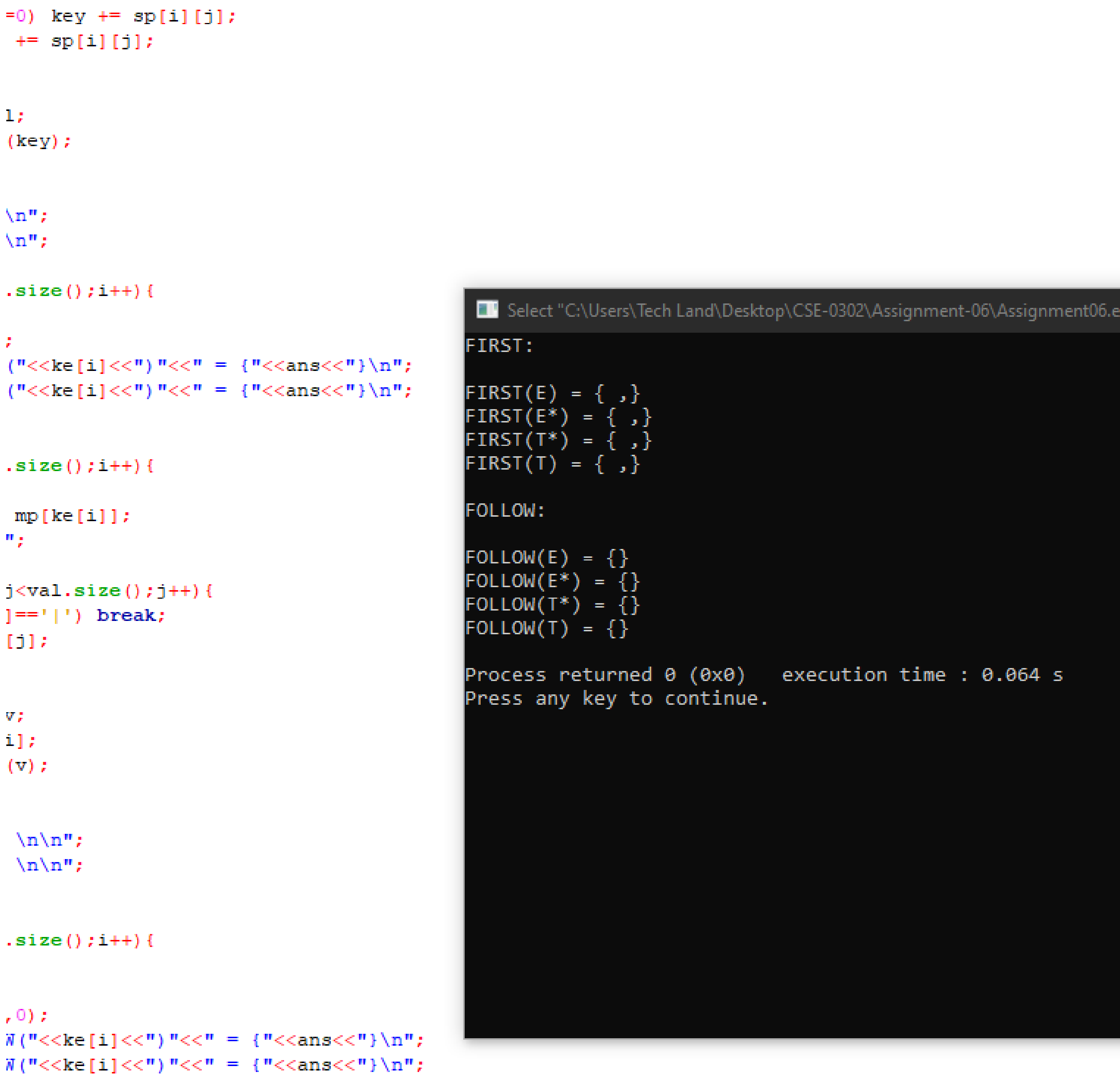
```
}



int main(){

    freopen("input.txt","r",stdin);
    freopen("out.txt","w",stdout);

    string s;

    while(getline(cin,s)){
        sp.push_back(remove_space(s));
    }

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

        string key="",val="";

        for(int j=0;j<sp[i].size();j++){
            if(sp[i][j]=='='){
                flag = 1;
                continue;
            }

            if(flag==0) key += sp[i][j];
            else val += sp[i][j];
        }

        mp[key] = val;
        ke.push_back(key);
    }

    cerr<<"FIRST: \n\n";
    cout<<"FIRST: \n\n";

    for(int i=0;i<ke.size();i++){
        ans = "";
        FIRST(ke[i]);
        cerr<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
        cout<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
    }

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

        string val = mp[ke[i]];
        string v = "";

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

Fig. 14. Assignment 6 Page 3



```
                if(val[j]=='|') break;
                v += val[j];
            }

            mp[ke[i]] = v;
            mpp[v] = ke[i];
            ri.push_back(v);
        }

        cerr<<"\nFOLLOW: \n\n";
        cout<<"\nFOLLOW: \n\n";


        for(int i=0;i<ke.size();i++){
            ans = "";

            FOLLOW(ke[i],0);
            cerr<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
            cout<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
        }



    }
```

Fig. 15. Assignment 6 Page 4



```
=0) key += sp[i][j];
 += sp[i][j];


l;
(key);


\n";
\n";

.size();i++){

;
("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
("<<ke[i]<<")"<<" = {"<<ans<<"}\n";


.size();i++){

 mp[ke[i]];
";

j<val.size();j++){
]=='|') break;
[j];


v;
i];
(v);


 \n\n";
 \n\n";


.size();i++){


,0);
W("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
W("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
```

Select "C:\Users\Tech Land\Desktop\CSE-0302\Assignment-06\Assignment06.e

```
FIRST:

FIRST(E) = { ,}
FIRST(E*) = { ,}
FIRST(T*) = { ,}
FIRST(T) = { ,}

FOLLOW:

FOLLOW(E) = {}
FOLLOW(E*) = {}
FOLLOW(T*) = {}
FOLLOW(T) = {}

Process returned 0 (0x0)   execution time : 0.064 s
Press any key to continue.
```

Fig. 16. Assignment 6 Page 5