```
#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){
        sp1.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;
            sp1.push_back(str);
            continue;
        }
        if(flag2){
            sp1.push_back(str);
            continue;
        }
        str = str + " " + sp[i];
        sp1.push_back(str);
    }

    return sp1;

}
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]=='}'){
                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;
    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;
        }
    }
    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";
    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;
}
```