0 – Cover Sheet

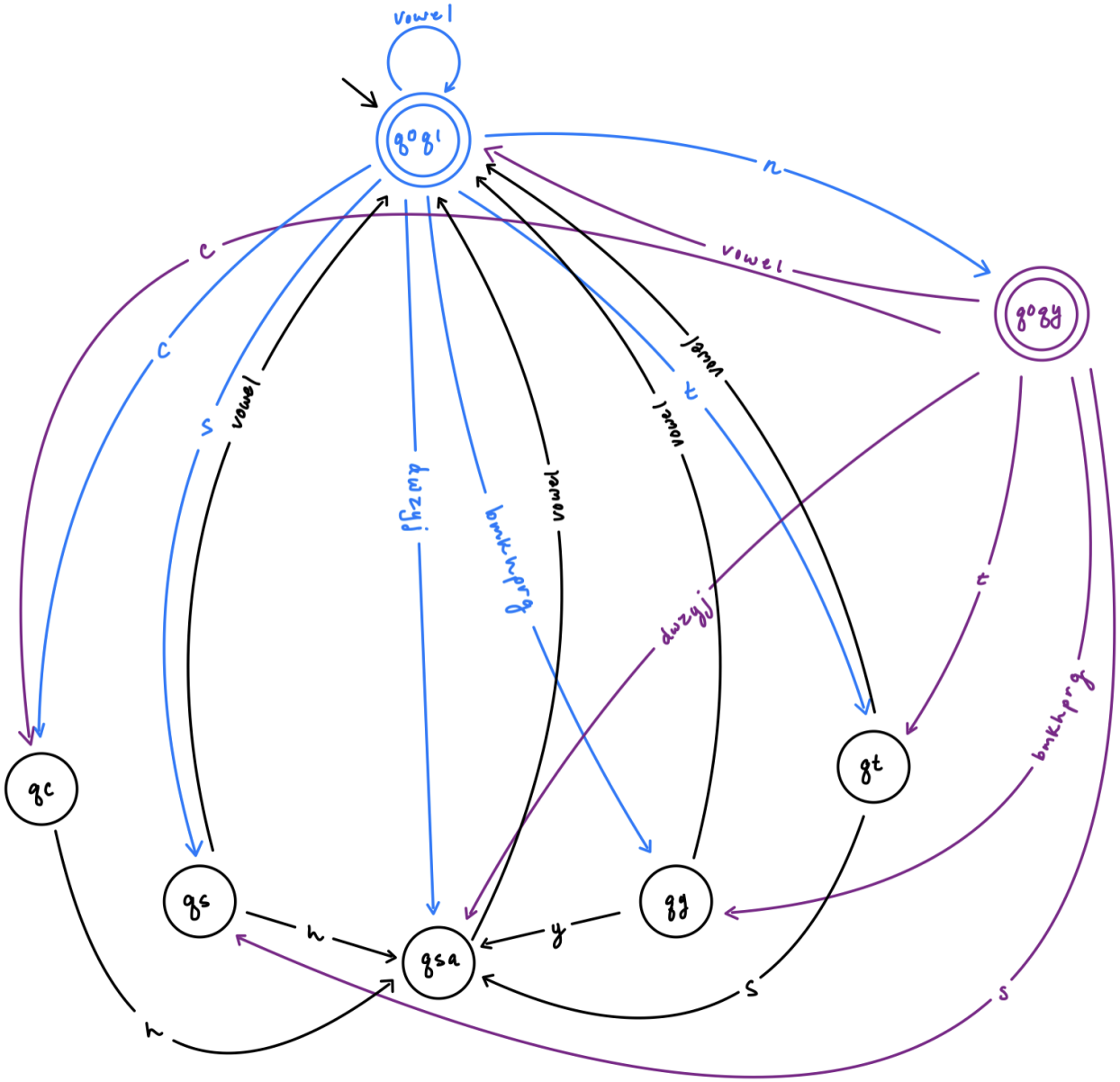
Japanese Scanner/Parser/Translator Project

Group 23
Leouel Guanzon
Marco Flores
John Foster

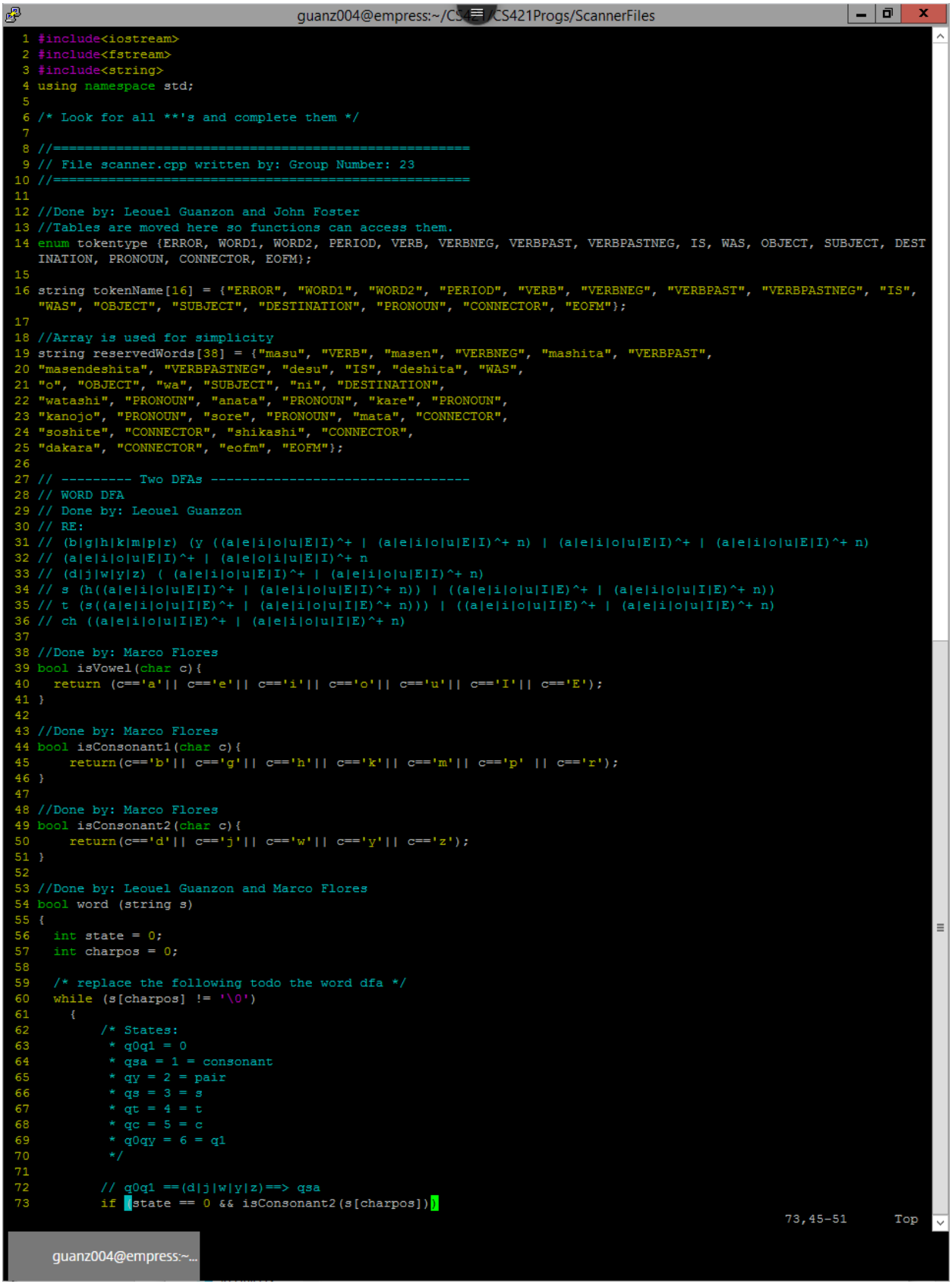
State of the program:

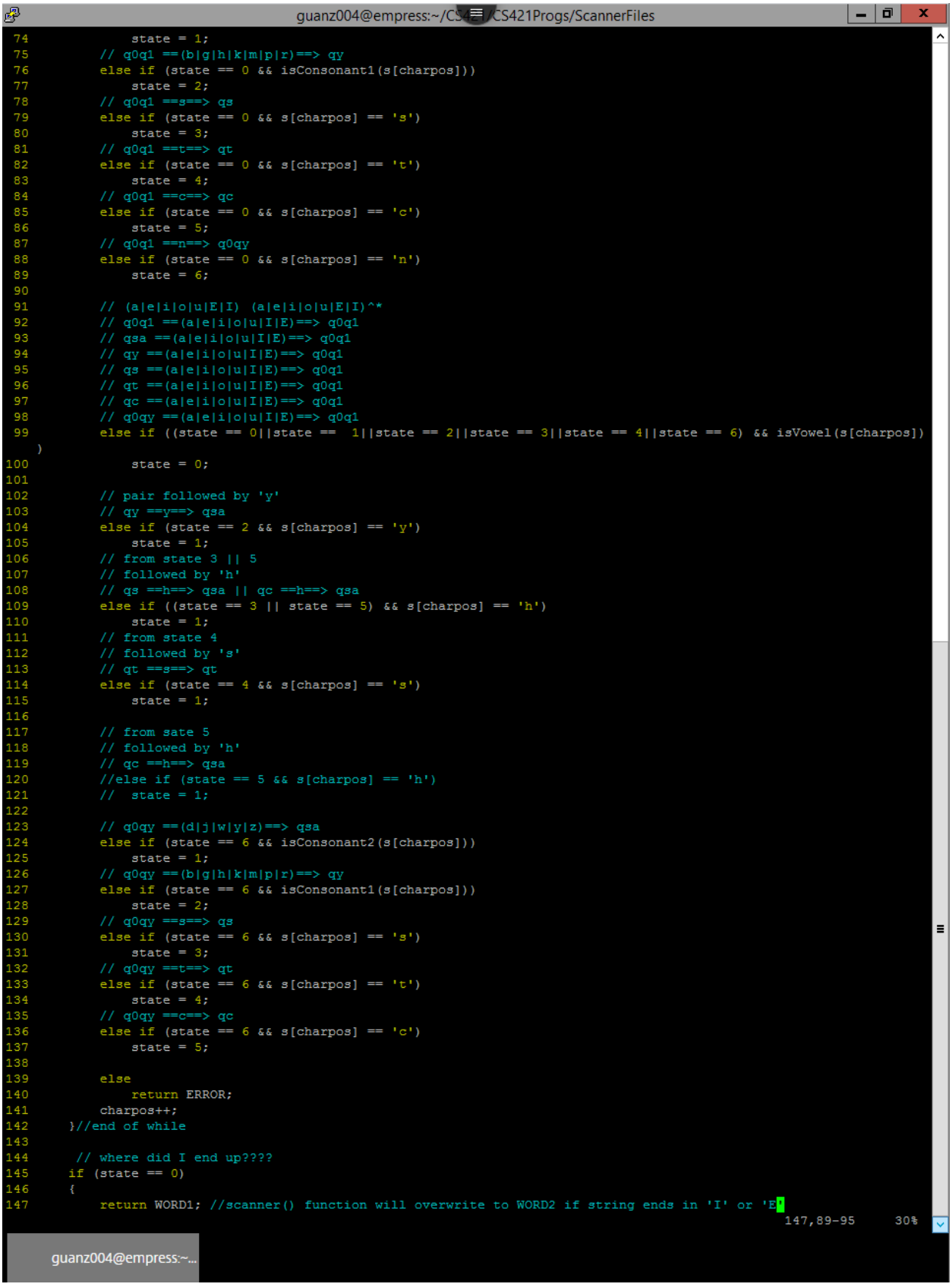
- Program works perfectly.
- All parts are completed.
- There are no bugs, and the program works as intended.
- No Extra Credit features implemented.

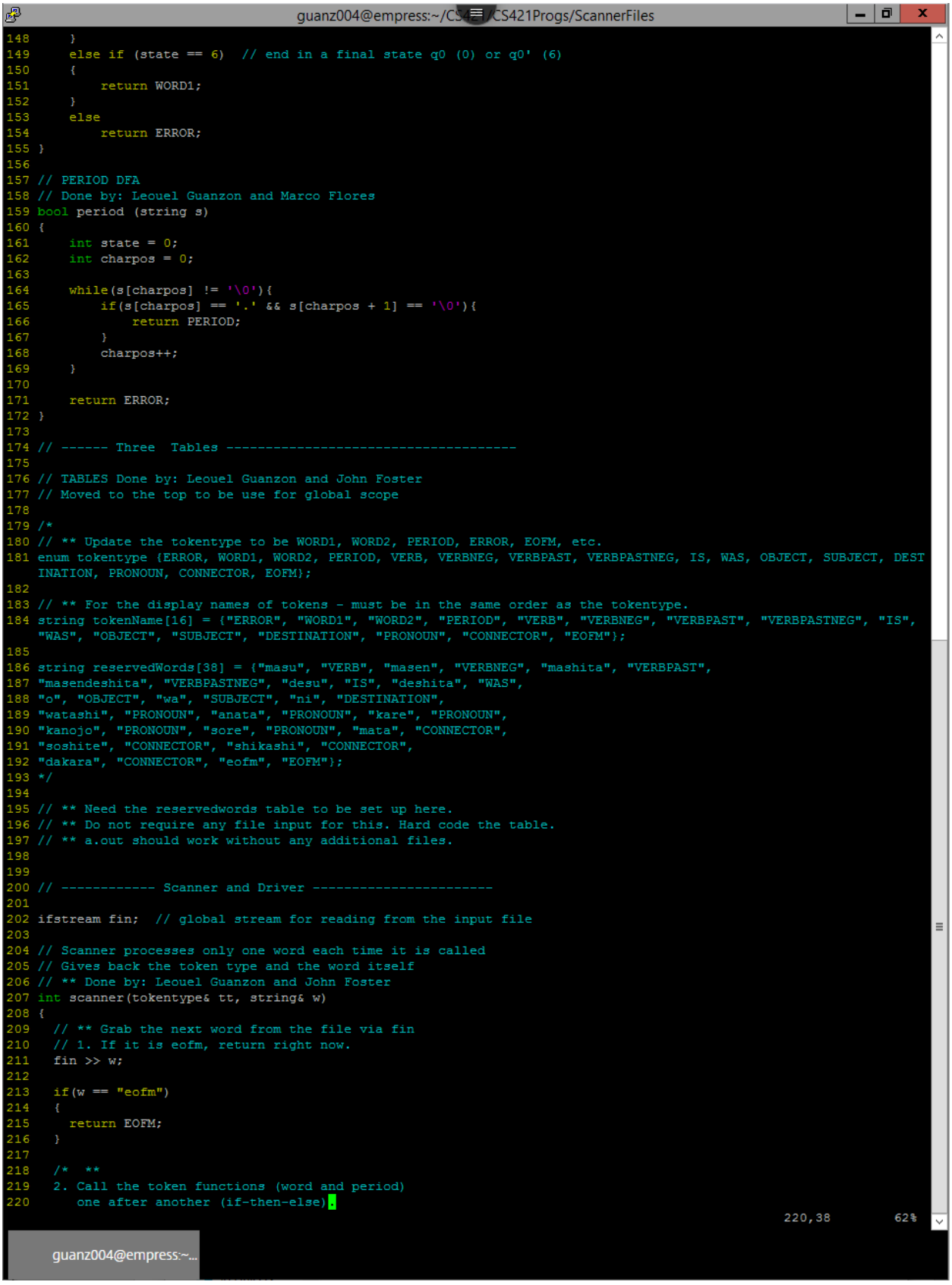
1 – DFA

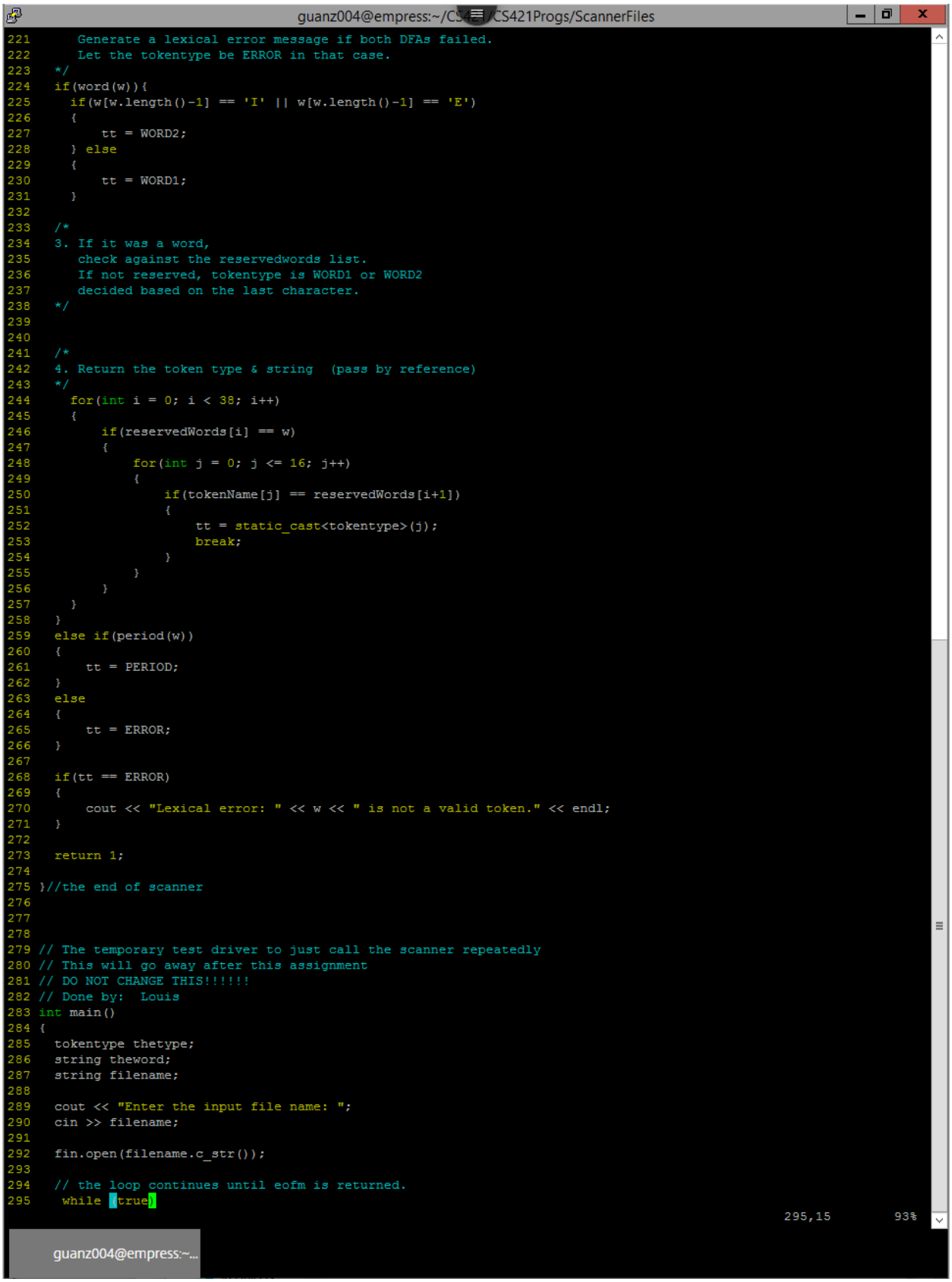

vowel
q0q1
n
c
vowel
q0q2
c
s
vowel
dwzyj
bmklwprq
vowel
vowel
t
dwzyj
t
bmklwprq
qc
qs
qj
qt
h
y
qsa
h
s
s

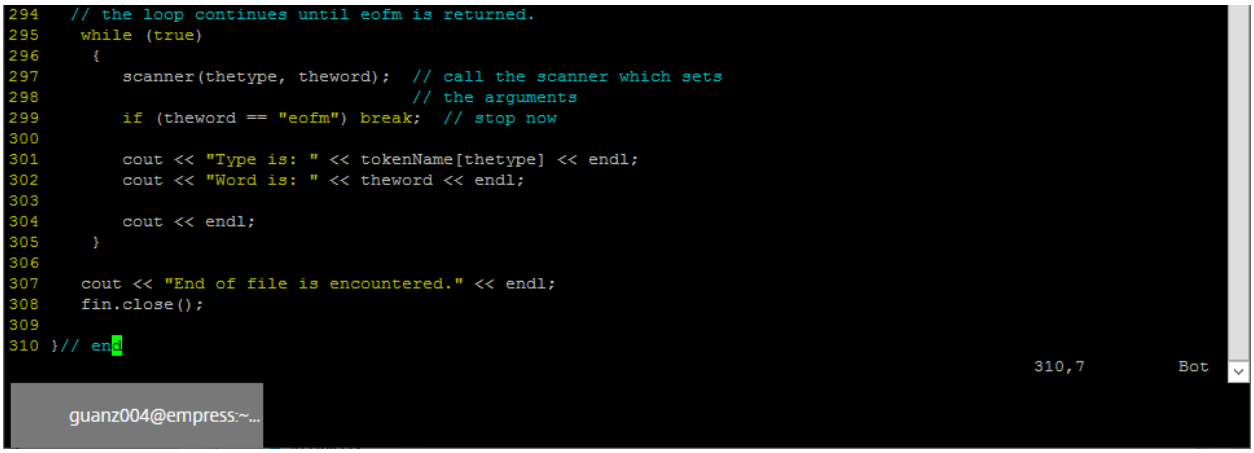
2 – Scanner.cpp



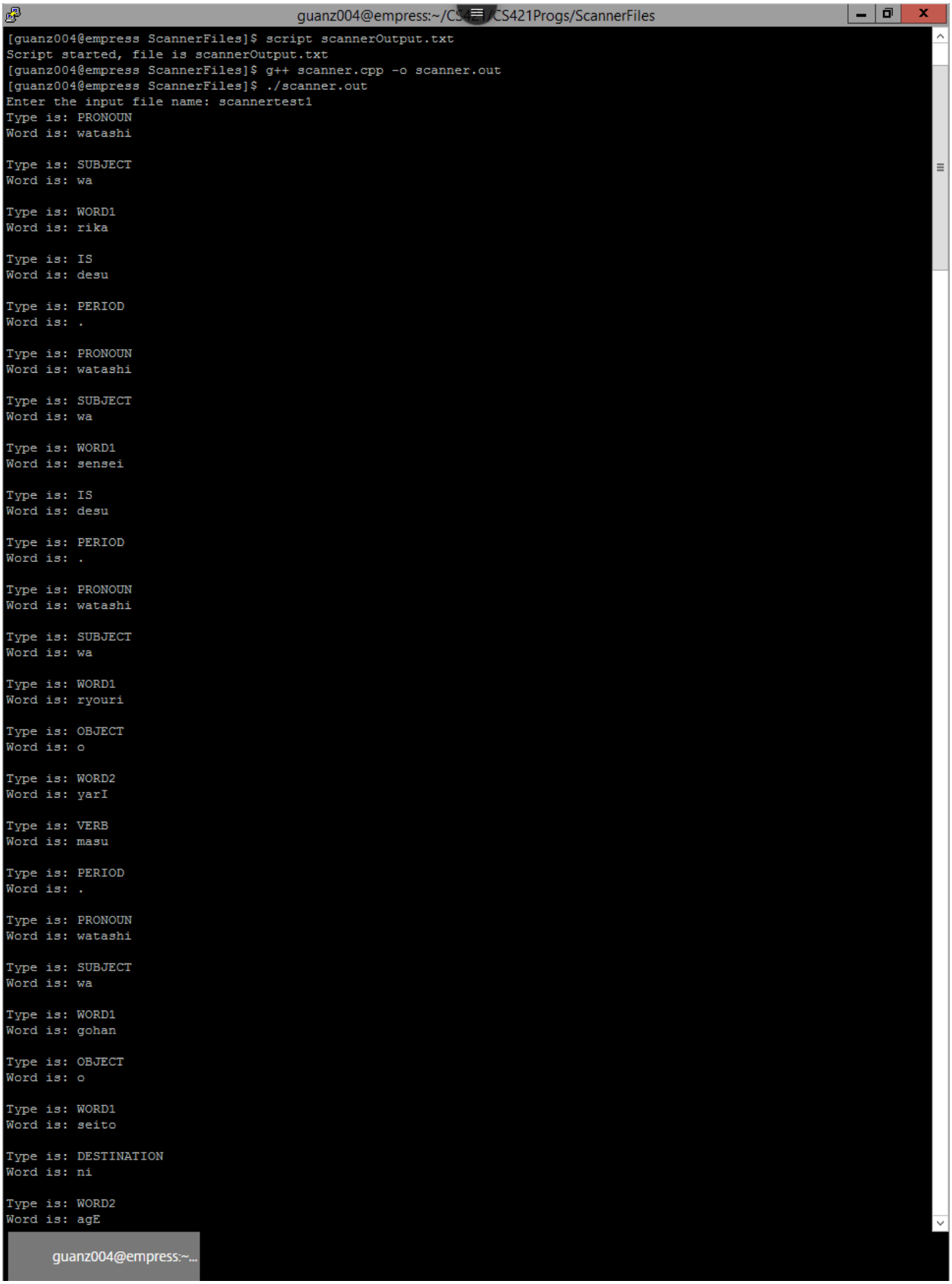

guanz004@empress:~/CS421Progs/ScannerFiles
74 state = 1;
75 // q0q1 ==(b|g|h|k|m|p|r)==> qy
76 else if (state == 0 && isConsonant1(s[charpos]))
77 state = 2;
78 // q0q1 ==s==> qs
79 else if (state == 0 && s[charpos] == 's')
80 state = 3;
81 // q0q1 ==t==> qt
82 else if (state == 0 && s[charpos] == 't')
83 state = 4;
84 // q0q1 ==c==> qc
85 else if (state == 0 && s[charpos] == 'c')
86 state = 5;
87 // q0q1 ==n==> q0qy
88 else if (state == 0 && s[charpos] == 'n')
89 state = 6;
90
91 // (a|e|i|o|u|E|I) (a|e|i|o|u|E|I)^*
92 // q0q1 ==(a|e|i|o|u|I|E)==> q0q1
93 // qsa ==(a|e|i|o|u|I|E)==> q0q1
94 // qy ==(a|e|i|o|u|I|E)==> q0q1
95 // qs ==(a|e|i|o|u|I|E)==> q0q1
96 // qt ==(a|e|i|o|u|I|E)==> q0q1
97 // qc ==(a|e|i|o|u|I|E)==> q0q1
98 // q0qy ==(a|e|i|o|u|I|E)==> q0q1
99 else if ((state == 0||state == 1||state == 2||state == 3||state == 4||state == 6) && isVowel(s[charpos])
)
100 state = 0;
101
102 // pair followed by 'y'
103 // qy ==y==> qsa
104 else if (state == 2 && s[charpos] == 'y')
105 state = 1;
106 // from state 3 || 5
107 // followed by 'h'
108 // qs ==h==> qsa || qc ==h==> qsa
109 else if ((state == 3 || state == 5) && s[charpos] == 'h')
110 state = 1;
111 // from state 4
112 // followed by 's'
113 // qt ==s==> qt
114 else if (state == 4 && s[charpos] == 's')
115 state = 1;
116
117 // from sate 5
118 // followed by 'h'
119 // qc ==h==> qsa
120 //else if (state == 5 && s[charpos] == 'h')
121 // state = 1;
122
123 // q0qy ==(d|j|w|y|z)==> qsa
124 else if (state == 6 && isConsonant2(s[charpos]))
125 state = 1;
126 // q0qy ==(b|g|h|k|m|p|r)==> qy
127 else if (state == 6 && isConsonant1(s[charpos]))
128 state = 2;
129 // q0qy ==s==> qs
130 else if (state == 6 && s[charpos] == 's')
131 state = 3;
132 // q0qy ==t==> qt
133 else if (state == 6 && s[charpos] == 't')
134 state = 4;
135 // q0qy ==c==> qc
136 else if (state == 6 && s[charpos] == 'c')
137 state = 5;
138
139 else
140 return ERROR;
141 charpos++;
142 }//end of while
143
144 // where did I end up????
145 if (state == 0)
146 {
147 return WORD1; //scanner() function will overwrite to WORD2 if string ends in 'I' or 'E'
147,89-95 30%
guanz004@empress:~...

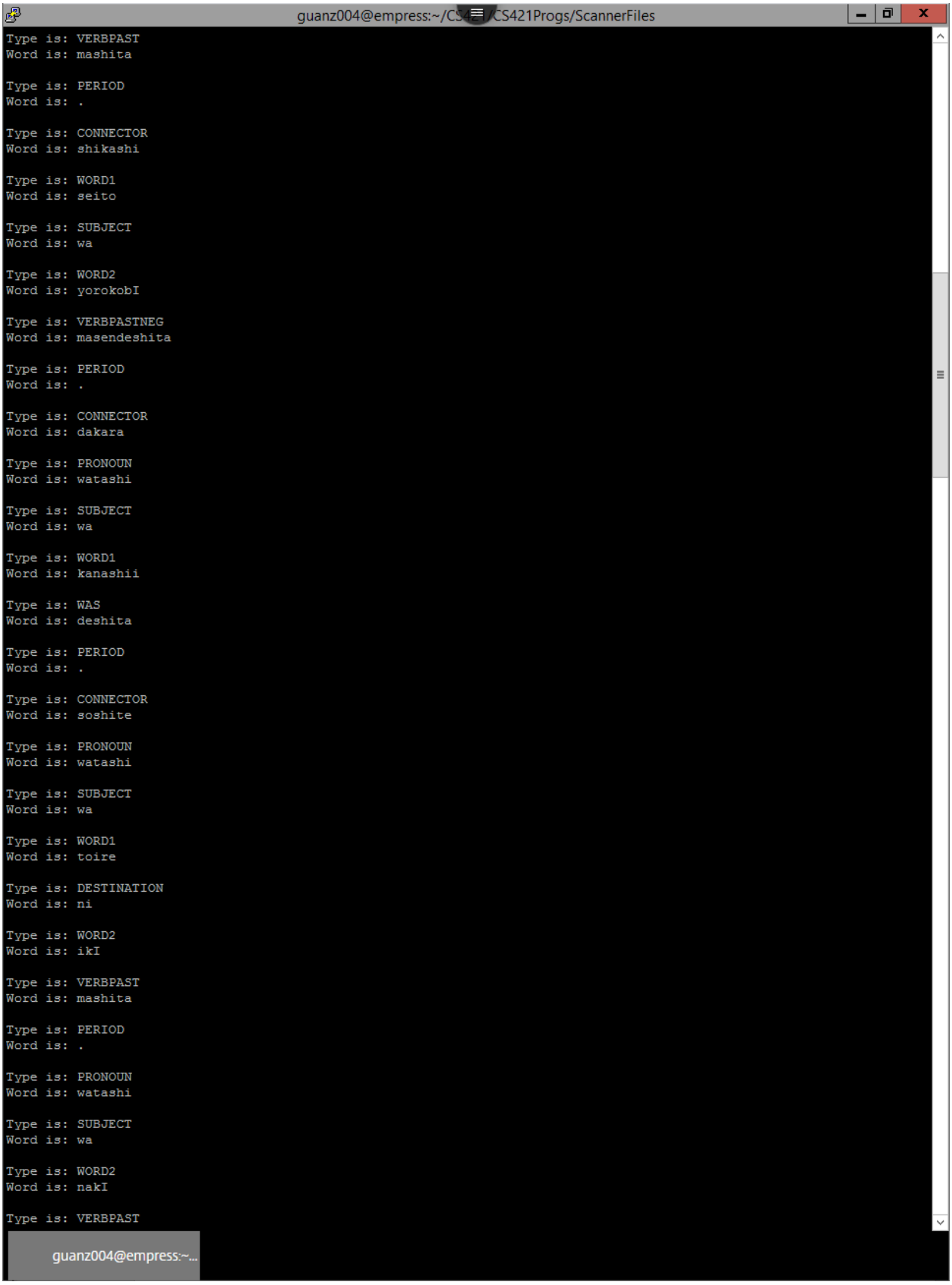

guanz004@empress:~/CS421Progs/ScannerFiles
148 }
149 else if (state == 6) // end in a final state q0 (0) or q0' (6)
150 {
151 return WORD1;
152 }
153 else
154 return ERROR;
155 }
156
157 // PERIOD DFA
158 // Done by: Leouel Guanzon and Marco Flores
159 bool period (string s)
160 {
161 int state = 0;
162 int charpos = 0;
163
164 while(s[charpos] != '\0'){
165 if(s[charpos] == '.' && s[charpos + 1] == '\0'){
166 return PERIOD;
167 }
168 charpos++;
169 }
170
171 return ERROR;
172 }
173
174 // ------ Three Tables ------------------------------------
175
176 // TABLES Done by: Leouel Guanzon and John Foster
177 // Moved to the top to be use for global scope
178
179 /*
180 // ** Update the tokentype to be WORD1, WORD2, PERIOD, ERROR, EOFM, etc.
181 enum tokentype {ERROR, WORD1, WORD2, PERIOD, VERB, VERBNEG, VERBPAST, VERBPASTNEG, IS, WAS, OBJECT, SUBJECT, DEST
INATION, PRONOUN, CONNECTOR, EOFM};
182
183 // ** For the display names of tokens - must be in the same order as the tokentype.
184 string tokenName[16] = {"ERROR", "WORD1", "WORD2", "PERIOD", "VERB", "VERBNEG", "VERBPAST", "VERBPASTNEG", "IS",
"WAS", "OBJECT", "SUBJECT", "DESTINATION", "PRONOUN", "CONNECTOR", "EOFM"};
185
186 string reservedWords[38] = {"masu", "VERB", "masen", "VERBNEG", "mashita", "VERBPAST",
187 "masendeshita", "VERBPASTNEG", "desu", "IS", "deshita", "WAS",
188 "o", "OBJECT", "wa", "SUBJECT", "ni", "DESTINATION",
189 "watashi", "PRONOUN", "anata", "PRONOUN", "kare", "PRONOUN",
190 "kanojo", "PRONOUN", "sore", "PRONOUN", "mata", "CONNECTOR",
191 "soshite", "CONNECTOR", "shikashi", "CONNECTOR",
192 "dakara", "CONNECTOR", "eofm", "EOFM"};
193 */
194
195 // ** Need the reservedwords table to be set up here.
196 // ** Do not require any file input for this. Hard code the table.
197 // ** a.out should work without any additional files.
198
199
200 // ------------ Scanner and Driver ----------------------
201
202 ifstream fin; // global stream for reading from the input file
203
204 // Scanner processes only one word each time it is called
205 // Gives back the token type and the word itself
206 // ** Done by: Leouel Guanzon and John Foster
207 int scanner(tokentype& tt, string& w)
208 {
209 // ** Grab the next word from the file via fin
210 // 1. If it is eofm, return right now.
211 fin >> w;
212
213 if(w == "eofm")
214 {
215 return EOFM;
216 }
217
218 /* **
219 2. Call the token functions (word and period)
220 one after another (if-then-else).
220,38 62%
guanz004@empress:~...

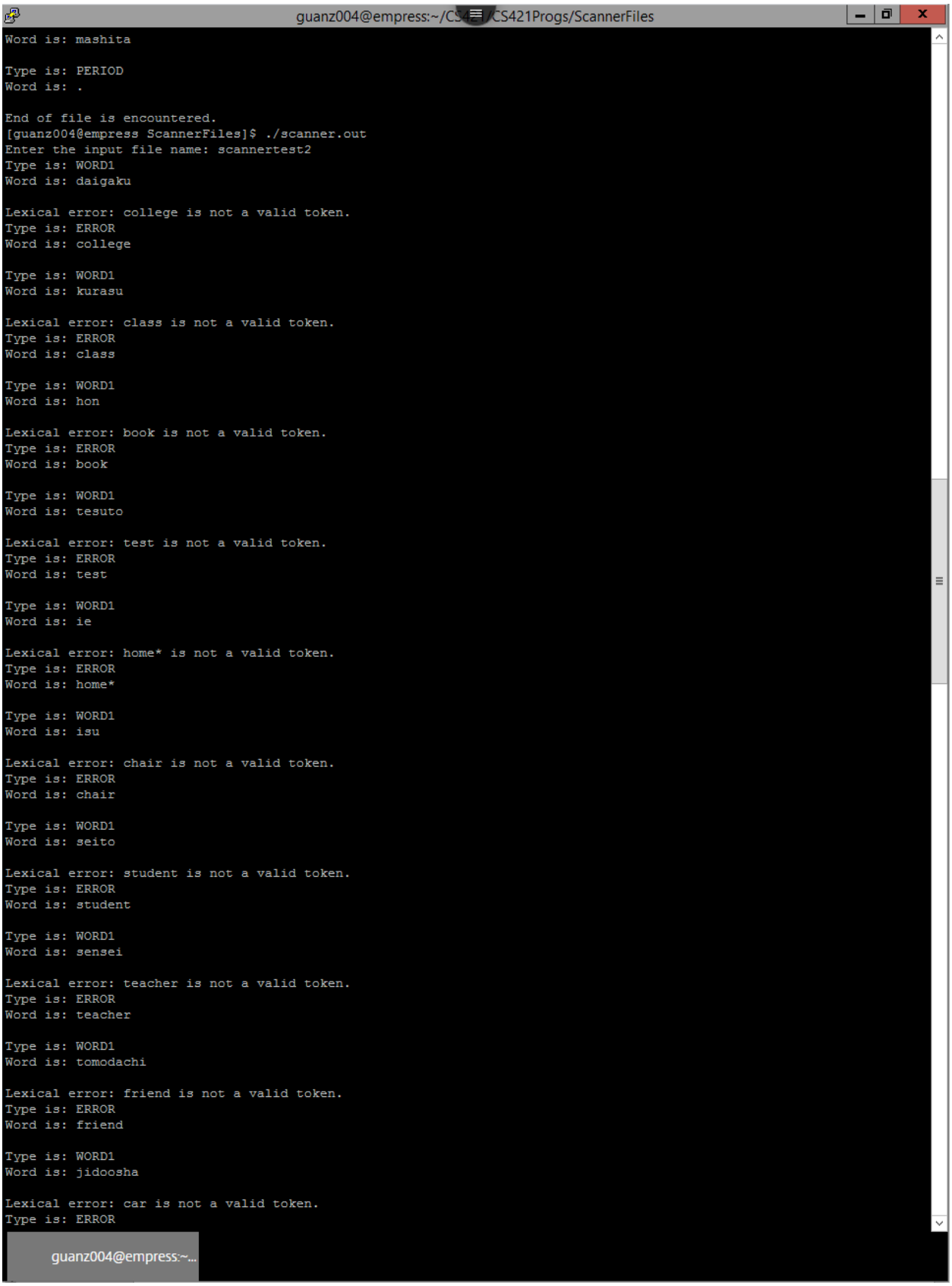

guanz004@empress:~/CS421/CS421Progs/ScannerFiles
Generate a lexical error message if both DFAs failed.
Let the tokentype be ERROR in that case.
*/
if(word(w)){
if(w[w.length()-1] == 'I' || w[w.length()-1] == 'E')
{
tt = WORD2;
} else
{
tt = WORD1;
}
/*
3. If it was a word,
check against the reservedwords list.
If not reserved, tokentype is WORD1 or WORD2
decided based on the last character.
*/
/*
4. Return the token type & string (pass by reference)
*/
for(int i = 0; i < 38; i++)
{
if(reservedWords[i] == w)
{
for(int j = 0; j <= 16; j++)
{
if(tokenName[j] == reservedWords[i+1])
{
tt = static_cast<tokentype>(j);
break;
}
}
}
}
}
else if(period(w))
{
tt = PERIOD;
}
else
{
tt = ERROR;
}
if(tt == ERROR)
{
cout << "Lexical error: " << w << " is not a valid token." << endl;
}
return 1;
}//the end of scanner
// The temporary test driver to just call the scanner repeatedly
// This will go away after this assignment
// DO NOT CHANGE THIS!!!!!!
// Done by: Louis
int main()
{
tokentype thetype;
string theword;
string filename;
cout << "Enter the input file name: ";
cin >> filename;
fin.open(filename.c_str());
// the loop continues until eofm is returned.
while (true)
295,15 93%
guanz004@empress:~...

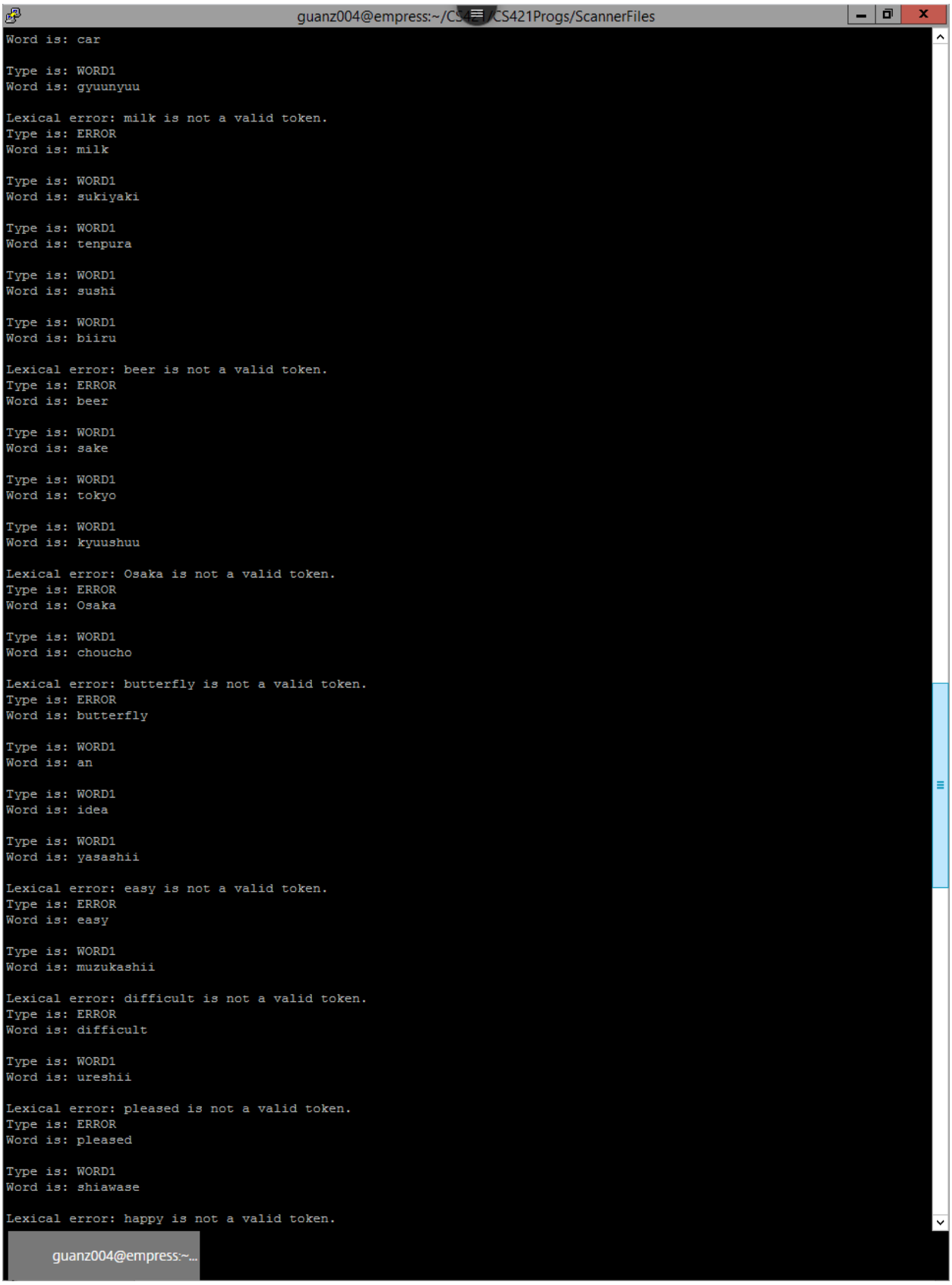

294 // the loop continues until eofm is returned.
295 while (true)
296 {
297 scanner(thetype, theword); // call the scanner which sets
298 // the arguments
299 if (theword == "eofm") break; // stop now
300
301 cout << "Type is: " << tokenName[thetype] << endl;
302 cout << "Word is: " << theword << endl;
303
304 cout << endl;
305 }
306
307 cout << "End of file is encountered." << endl;
308 fin.close();
309
310 }// end
310,7 Bot
guanz004@empress:~...

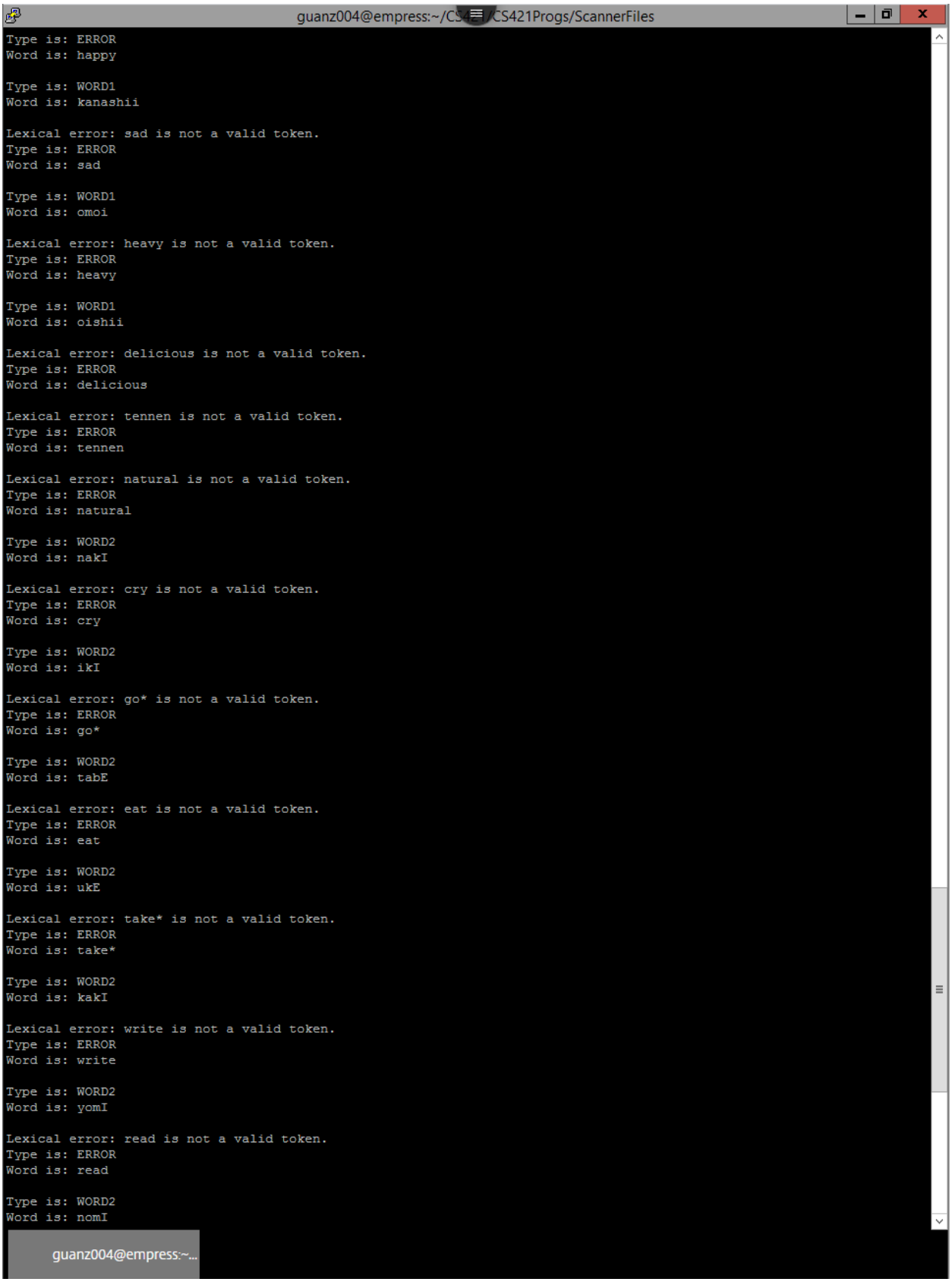
3 – Original Scanner test results

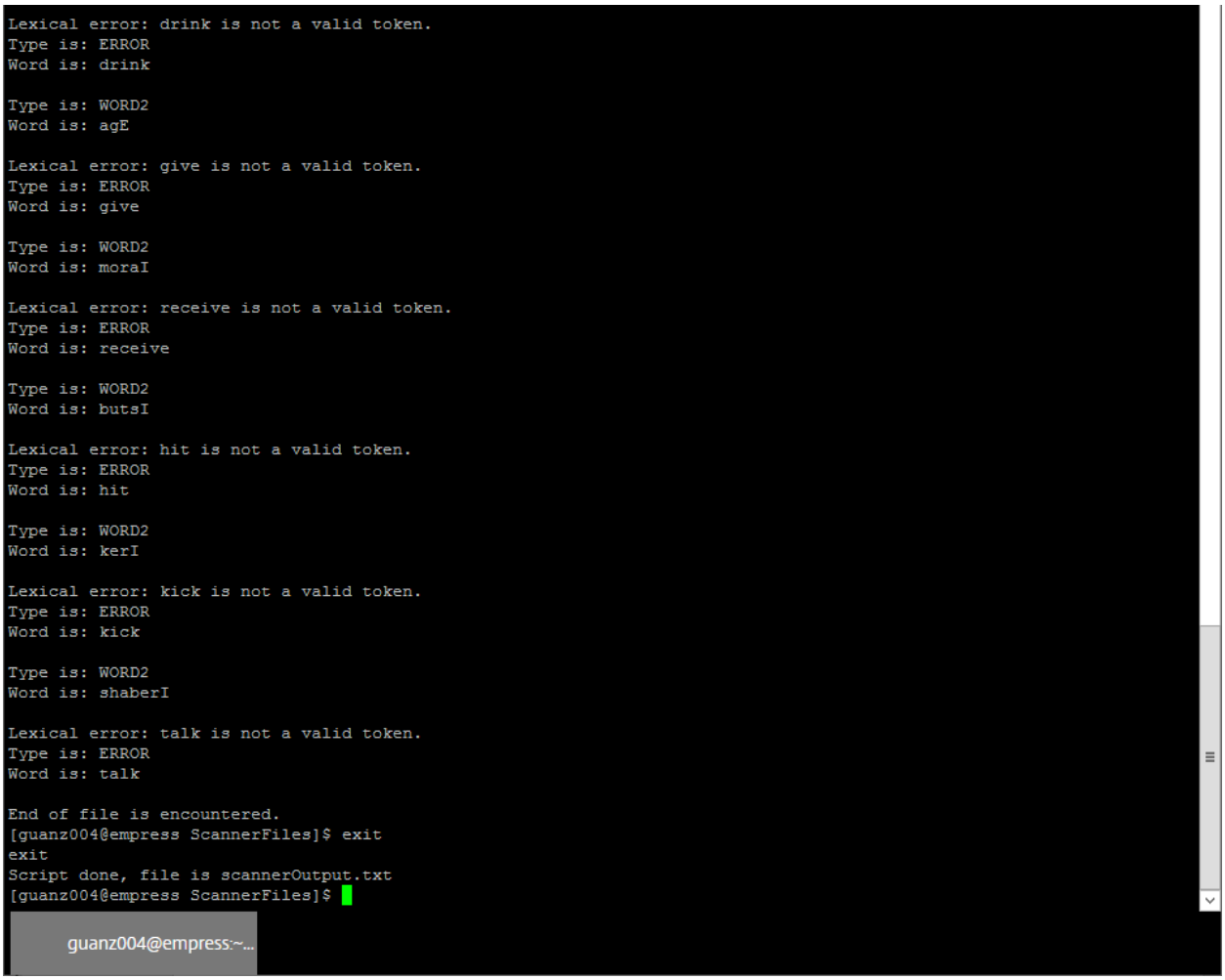



guanz004@empress:~/CS421Progs/ScannerFiles
Type is: VERBPAST
Word is: mashita
Type is: PERIOD
Word is: .
Type is: CONNECTOR
Word is: shikashi
Type is: WORD1
Word is: seito
Type is: SUBJECT
Word is: wa
Type is: WORD2
Word is: yorokobI
Type is: VERBPASTNEG
Word is: masendeshita
Type is: PERIOD
Word is: .
Type is: CONNECTOR
Word is: dakara
Type is: PRONOUN
Word is: watashi
Type is: SUBJECT
Word is: wa
Type is: WORD1
Word is: kanashii
Type is: WAS
Word is: deshita
Type is: PERIOD
Word is: .
Type is: CONNECTOR
Word is: soshite
Type is: PRONOUN
Word is: watashi
Type is: SUBJECT
Word is: wa
Type is: WORD1
Word is: toire
Type is: DESTINATION
Word is: ni
Type is: WORD2
Word is: ikI
Type is: VERBPAST
Word is: mashita
Type is: PERIOD
Word is: .
Type is: PRONOUN
Word is: watashi
Type is: SUBJECT
Word is: wa
Type is: WORD2
Word is: nakI
Type is: VERBPAST
guanz004@empress:~...


guanz004@empress:~/CS421Progs/ScannerFiles
Word is: mashita
Type is: PERIOD
Word is: .
End of file is encountered.
[guanz004@empress ScannerFiles]$ ./scanner.out
Enter the input file name: scannertest2
Type is: WORD1
Word is: daigaku
Lexical error: college is not a valid token.
Type is: ERROR
Word is: college
Type is: WORD1
Word is: kurasu
Lexical error: class is not a valid token.
Type is: ERROR
Word is: class
Type is: WORD1
Word is: hon
Lexical error: book is not a valid token.
Type is: ERROR
Word is: book
Type is: WORD1
Word is: tesuto
Lexical error: test is not a valid token.
Type is: ERROR
Word is: test
Type is: WORD1
Word is: ie
Lexical error: home* is not a valid token.
Type is: ERROR
Word is: home*
Type is: WORD1
Word is: isu
Lexical error: chair is not a valid token.
Type is: ERROR
Word is: chair
Type is: WORD1
Word is: seito
Lexical error: student is not a valid token.
Type is: ERROR
Word is: student
Type is: WORD1
Word is: sensei
Lexical error: teacher is not a valid token.
Type is: ERROR
Word is: teacher
Type is: WORD1
Word is: tomodachi
Lexical error: friend is not a valid token.
Type is: ERROR
Word is: friend
Type is: WORD1
Word is: jidoosha
Lexical error: car is not a valid token.
Type is: ERROR
guanz004@empress:~...


guanz004@empress:~/CS421/CS421Progs/ScannerFiles
Word is: car
Type is: WORD1
Word is: gyuunyuu
Lexical error: milk is not a valid token.
Type is: ERROR
Word is: milk
Type is: WORD1
Word is: sukiyaki
Type is: WORD1
Word is: tenpura
Type is: WORD1
Word is: sushi
Type is: WORD1
Word is: biiru
Lexical error: beer is not a valid token.
Type is: ERROR
Word is: beer
Type is: WORD1
Word is: sake
Type is: WORD1
Word is: tokyo
Type is: WORD1
Word is: kyuushuu
Lexical error: Osaka is not a valid token.
Type is: ERROR
Word is: Osaka
Type is: WORD1
Word is: choucho
Lexical error: butterfly is not a valid token.
Type is: ERROR
Word is: butterfly
Type is: WORD1
Word is: an
Type is: WORD1
Word is: idea
Type is: WORD1
Word is: yasashii
Lexical error: easy is not a valid token.
Type is: ERROR
Word is: easy
Type is: WORD1
Word is: muzukashii
Lexical error: difficult is not a valid token.
Type is: ERROR
Word is: difficult
Type is: WORD1
Word is: ureshii
Lexical error: pleased is not a valid token.
Type is: ERROR
Word is: pleased
Type is: WORD1
Word is: shiawase
Lexical error: happy is not a valid token.
guanz004@empress:~...


guanz004@empress:~/CS421Progs/ScannerFiles
Type is: ERROR
Word is: happy
Type is: WORD1
Word is: kanashii
Lexical error: sad is not a valid token.
Type is: ERROR
Word is: sad
Type is: WORD1
Word is: omoi
Lexical error: heavy is not a valid token.
Type is: ERROR
Word is: heavy
Type is: WORD1
Word is: oishii
Lexical error: delicious is not a valid token.
Type is: ERROR
Word is: delicious
Lexical error: tennen is not a valid token.
Type is: ERROR
Word is: tennen
Lexical error: natural is not a valid token.
Type is: ERROR
Word is: natural
Type is: WORD2
Word is: nakI
Lexical error: cry is not a valid token.
Type is: ERROR
Word is: cry
Type is: WORD2
Word is: ikI
Lexical error: go* is not a valid token.
Type is: ERROR
Word is: go*
Type is: WORD2
Word is: tabE
Lexical error: eat is not a valid token.
Type is: ERROR
Word is: eat
Type is: WORD2
Word is: ukE
Lexical error: take* is not a valid token.
Type is: ERROR
Word is: take*
Type is: WORD2
Word is: kakI
Lexical error: write is not a valid token.
Type is: ERROR
Word is: write
Type is: WORD2
Word is: yomI
Lexical error: read is not a valid token.
Type is: ERROR
Word is: read
Type is: WORD2
Word is: nomI
guanz004@empress:~...


Lexical error: drink is not a valid token.
Type is: ERROR
Word is: drink
Type is: WORD2
Word is: agE
Lexical error: give is not a valid token.
Type is: ERROR
Word is: give
Type is: WORD2
Word is: moraI
Lexical error: receive is not a valid token.
Type is: ERROR
Word is: receive
Type is: WORD2
Word is: butsI
Lexical error: hit is not a valid token.
Type is: ERROR
Word is: hit
Type is: WORD2
Word is: kerI
Lexical error: kick is not a valid token.
Type is: ERROR
Word is: kick
Type is: WORD2
Word is: shaberI
Lexical error: talk is not a valid token.
Type is: ERROR
Word is: talk
End of file is encountered.
[guanz004@empress ScannerFiles]$ exit
exit
Script done, file is scannerOutput.txt
[guanz004@empress ScannerFiles]$
guanz004@empress:~...

4 – Factored Rules

1 <story> ::= <s> { <s> } // stay in the loop as long as a possible start
// of <s> is the next_token (note it can be CONNECTOR or WORD1 or PRONOUN)

2 <s> ::= [CONNECTOR] <noun> SUBJECT <verb> <tense> PERIOD
3 <s> ::= [CONNECTOR] <noun> SUBJECT <noun> <be> PERIOD
4 <s> ::= [CONNECTOR] <noun> SUBJECT <noun> DESTINATION <verb> <tense> PERIOD
5 <s> ::= [CONNECTOR] <noun> SUBJECT <noun> OBJECT <verb> <tense> PERIOD
6 <s> ::= [CONNECTOR] <noun> SUBJECT <noun> OBJECT <noun> DESTINATION <verb> <tense> PERIOD

// Refer to Left Factoring file to make these into
// one rule until things start to differ.

<s> ::= [CONNECTOR #getEword# #gen(CONNECTOR)#]
<noun> #getEword# SUBJECT #gen(ACTOR)#
<after subject>

<after subject> ::= <verb> #getEword# #gen(ACTION)#
<tense> #gen(TENSE)# PERIOD |
<noun> #getEword#
<after noun>

<after noun> ::= <be> #gen(DESCRIPTION)# #gen(TENSE)# PERIOD |
DESTINATION #gen(TO)#
<verb> #getEword# #gen(ACTION)#
<tense> #gen(TENSE)# PERIOD |
OBJECT #gen(OBJECT)#
<after object>

<after object> ::= <verb> #getEword# #gen(ACTION)#
<tense> #gen(TENSE)# PERIOD |
<noun> #getEword# DESTINATION #gen(TO)#
<verb> #getEword# #gen(ACTION)#
<tense> #gen(TENSE)# PERIOD

7 <noun> ::= WORD1 | PRONOUN

8 <verb> ::= WORD2

9 <be> ::= IS | WAS

10 <tense> := VERBPAST | VERBPASTNEG | VERB | VERBNEG

5 – Updated Parser code for Translation

guanz004@empress:~/CS[redacted]CS421Progs/TranslatorFiles

```
#include <iostream>
#include <fstream>
#include <string>
#include <unordered_map>
#include <stdlib.h>
#include <cstring>
#include "scanner.hpp"
using namespace std;

ofstream fout;
ofstream ferr;
string saved_lexeme;
tokentype saved_token;
bool token_available = false;
bool disable_tracing = false;
string token_Name[16] = {"ERROR", "WORD1", "WORD2", "PERIOD", "VERB", "VERBNEG", "VERBPAST", "VERBPASTNEG", "IS", "WAS"
, "OBJECT", "SUBJECT", "DESTINATION", "PRONOUN", "CONNECTOR", "EOFM"};

string filename;


/* INSTRUCTION:  copy your parser.cpp here
      cp ../ParserFiles/parser.cpp .
   Then, insert or append its contents into this file and edit.
   Complete all ** parts.
*/

/* INSTRUCTION:  Complete all ** parts.
   You may use any method to connect this file to scanner.cpp
   that you had written.
  e.g. You can copy scanner.cpp here by:
         cp ../ScannerFiles/scanner.cpp .
       and then append the two files into one:
         cat scanner.cpp parser.cpp > myparser.cpp
*/

//=================================================
// File translator.cpp written by Group Number: 23
//=================================================

// ----- Additions to the parser.cpp ---------------------

// ** Declare Lexicon (i.e. dictionary) that will hold the content of lexicon.txt
// Make sure it is easy and fast to look up the translation.
// Do not change the format or content of lexicon.txt
//  Done by: Marco Flores
unordered_map<string, string> lexicon_map;
string saved_E_word;


// ** Additions to parser.cpp here:
//    getEword() - using the current saved_lexeme, look up the English word
//                 in Lexicon if it is there -- save the result
//                 in saved_E_word
//  Done by: Marco Flores
bool getEword(){
  std::unordered_map<std::string,string>::const_iterator it = lexicon_map.find(saved_lexeme);
  if ( it == lexicon_map.end()) {
        //No translation exists, so we set the "english word" as just the original word in japanese
        saved_E_word = saved_lexeme;
        return 0;
  }
  //Found a translation, save it as the english word
  saved_E_word = it->second;
  return 1;
}

//    gen(line_type) - using the line type,
//                     sends a line of an IR to translated.txt
//                     (saved_E_word or saved_token is used)
//Done by: Leouel Guanzon
void gen(string line_type){
  //use fout, the file is already open
"translator.cpp" [dos] 465L, 10974C                                          1,1           Top
```

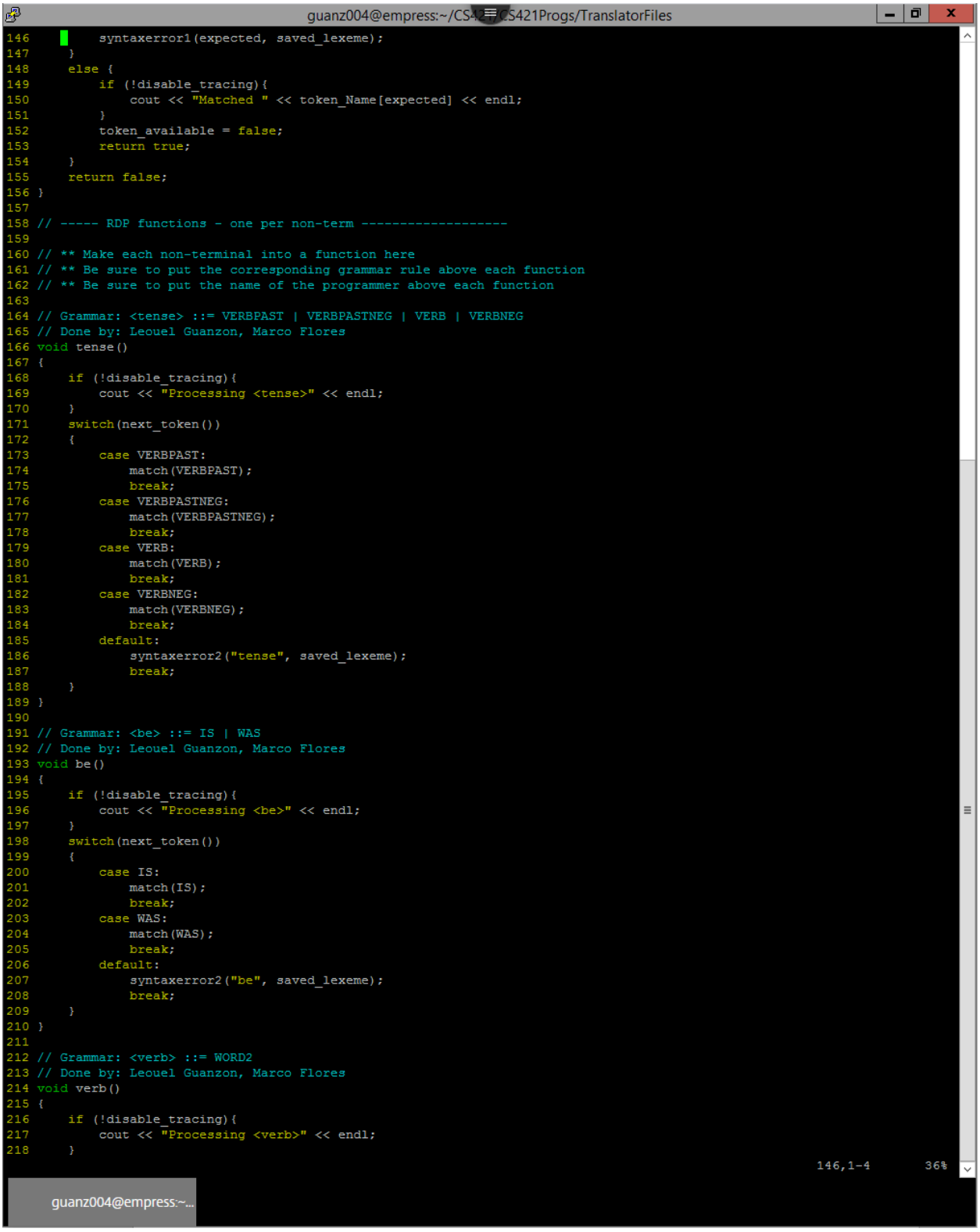
guanz004@empress:~...

guanz004@empress:~/CS4[redacted]/CS421Progs/TranslatorFiles

```
 73     if(line_type == "TENSE")
 74     {
 75         cout << line_type << ": " << token_Name[saved_token] << endl;
 76         fout << line_type << ": " << token_Name[saved_token] << endl;
 77     }
 78     else
 79     {
 80         cout << line_type << ": " << saved_E_word << endl;
 81         fout << line_type << ": " << saved_E_word << endl;
 82     }
 83 }
 84 // ----- Changes to the parser.cpp content --------------------
 85
 86 // ** Comment update: Be sure to put the corresponding grammar
 87 //    rule with semantic routine calls
 88 //    above each non-terminal function
 89
 90 // ** Each non-terminal function should be calling
 91 //    getEword and/or gen now.
 92
 93
 94 // Type of error: Match() fails
 95 // Done by: Leouel Guanzon
 96 void syntaxerror1(tokentype tt, string saved_lexeme)
 97 {
 98     cout << "\nSYNTAX ERROR: expected " << token_Name[tt] << " but found " << saved_lexeme << endl;
 99     ferr << "\nSYNTAX ERROR: expected " << token_Name[tt] << " but found " << saved_lexeme << endl;
100     exit(EXIT_FAILURE);
101 }
102
103 // Type of error: default of switch cases
104 // Done by: Leouel Guanzon
105 void syntaxerror2(string pFunction, string saved_lexeme)
106 {
107     cout << "\nSYNTAX ERROR: unexpected " << saved_lexeme << " found in " << pFunction << endl;
108     ferr << "\nSYNTAX ERROR: expected " << saved_lexeme << " but found " << pFunction << endl;
109     exit(EXIT_FAILURE);
110 }
111
112 // ** Need the updated match and next_token with 2 global vars
113 // saved_token and saved_lexeme
114
115 // Purpose: Looks at the next token from the scanner
116 // Done by: Leouel Guanzon
117 tokentype next_token()
118 {
119     if(!token_available)
120     {
121         scanner(saved_token, saved_lexeme);
122         cout << "Scanner called using word: " << saved_lexeme << endl;//not sure if this is a tracing message
123         if(saved_lexeme == "eofm")
124         {
125             cout << endl;
126             cout << "Successfully parsed <story>." << endl;
127             fout.close();
128             ferr.close();
129             exit(EXIT_SUCCESS);
130         }
131
132         token_available = true;
133     }
134
135     return saved_token;
136 }
137
138 // Purpose: Checks if token matches;
139 //          if true: returns true
140 //          if false: handles error using syntaxerror1()
141 // Done by: Leouel Guanzon
142 bool match(tokentype expected)
143 {
144     if(next_token() != expected)
145     {
```

73,1-4 18%

guanz004@empress:~...

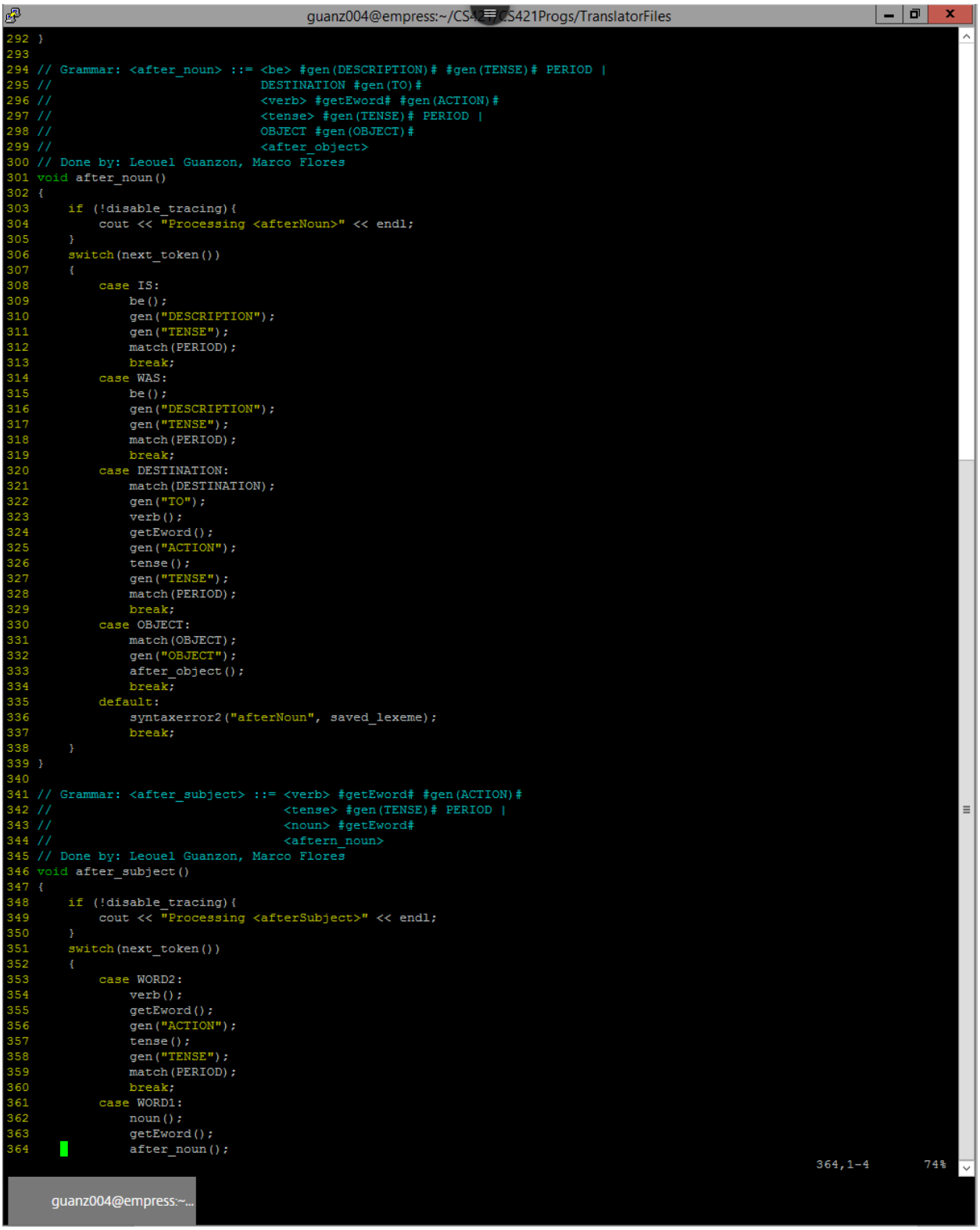

guanz004@empress:~/CS421Progs/TranslatorFiles
146 syntaxerror1(expected, saved_lexeme);
147 }
148 else {
149 if (!disable_tracing){
150 cout << "Matched " << token_Name[expected] << endl;
151 }
152 token_available = false;
153 return true;
154 }
155 return false;
156 }
157
158 // ----- RDP functions - one per non-term ------------------
159
160 // ** Make each non-terminal into a function here
161 // ** Be sure to put the corresponding grammar rule above each function
162 // ** Be sure to put the name of the programmer above each function
163
164 // Grammar: <tense> ::= VERBPAST | VERBPASTNEG | VERB | VERBNEG
165 // Done by: Leouel Guanzon, Marco Flores
166 void tense()
167 {
168 if (!disable_tracing){
169 cout << "Processing <tense>" << endl;
170 }
171 switch(next_token())
172 {
173 case VERBPAST:
174 match(VERBPAST);
175 break;
176 case VERBPASTNEG:
177 match(VERBPASTNEG);
178 break;
179 case VERB:
180 match(VERB);
181 break;
182 case VERBNEG:
183 match(VERBNEG);
184 break;
185 default:
186 syntaxerror2("tense", saved_lexeme);
187 break;
188 }
189 }
190
191 // Grammar: <be> ::= IS | WAS
192 // Done by: Leouel Guanzon, Marco Flores
193 void be()
194 {
195 if (!disable_tracing){
196 cout << "Processing <be>" << endl;
197 }
198 switch(next_token())
199 {
200 case IS:
201 match(IS);
202 break;
203 case WAS:
204 match(WAS);
205 break;
206 default:
207 syntaxerror2("be", saved_lexeme);
208 break;
209 }
210 }
211
212 // Grammar: <verb> ::= WORD2
213 // Done by: Leouel Guanzon, Marco Flores
214 void verb()
215 {
216 if (!disable_tracing){
217 cout << "Processing <verb>" << endl;
218 }
146,1-4 36%
guanz004@empress:~...

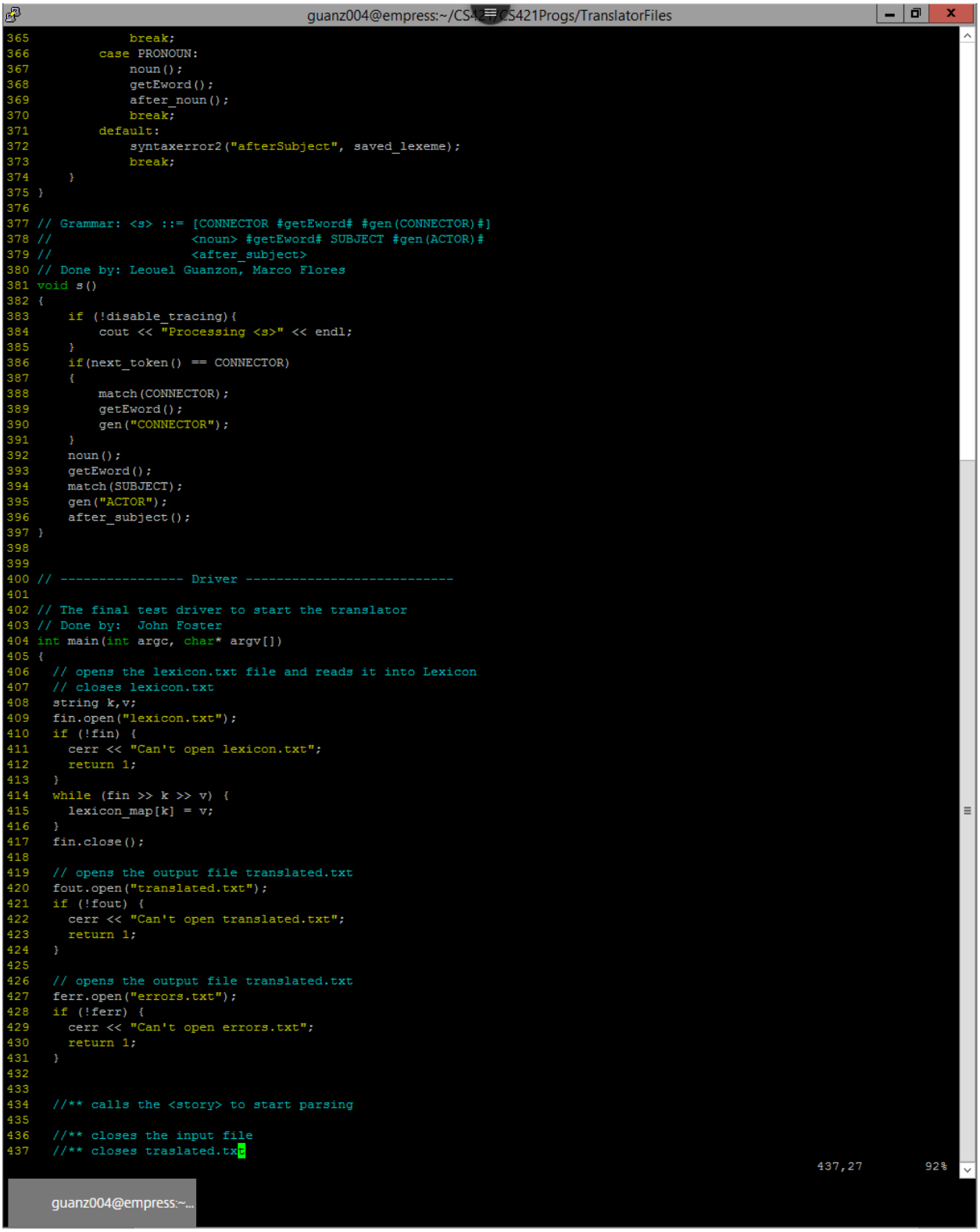
guanz004@empress:~/CS4[illegible]/CS421Progs/TranslatorFiles

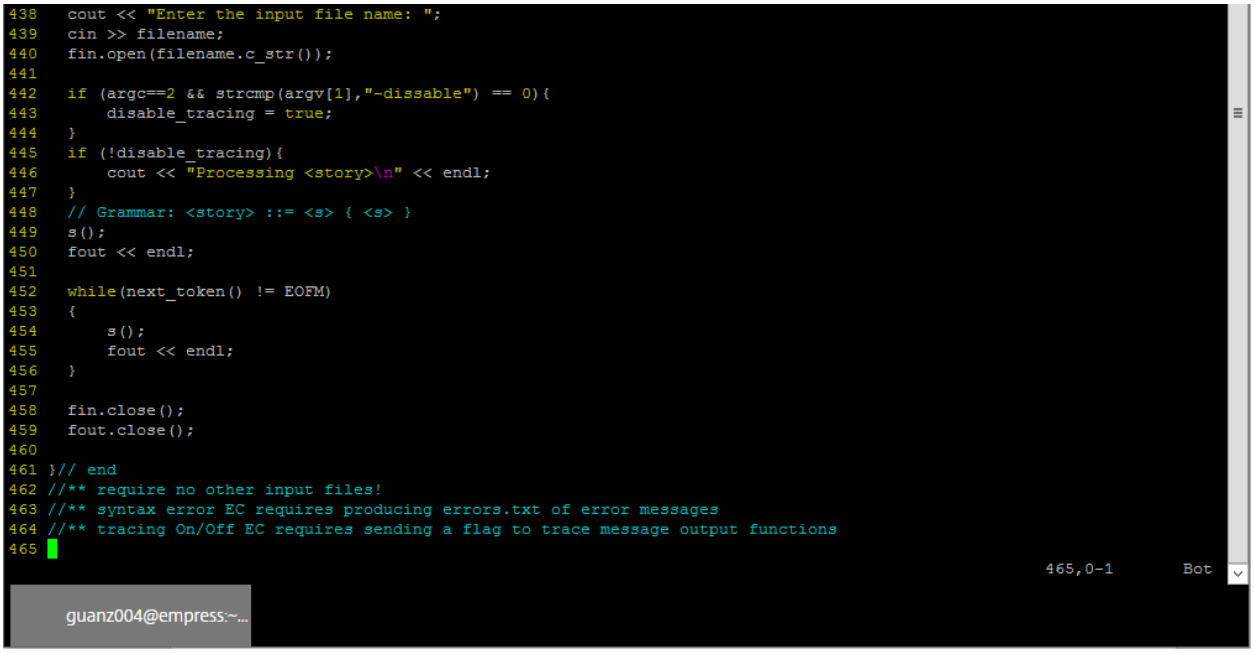
```
219     match(WORD2);
220 }
221
222 // Grammar: <noun> ::= WORD1 | PRONOUN
223 // Done by: Leouel Guanzon, Marco Flores
224 void noun()
225 {
226     if (!disable_tracing){
227         cout << "Processing <noun>" << endl;
228     }
229     switch(next_token())
230     {
231         case WORD1:
232             match(WORD1);
233             break;
234         case PRONOUN:
235             match(PRONOUN);
236             break;
237         default:
238             syntaxerror2("noun", saved_lexeme);
239             break;
240     }
241 }
242
243 // Grammar: <after_object> ::= <verb> #getEword# #gen(ACTION)#
244 //                             <tense> #gen(TENSE)# PERIOD |
245 //                             <noun> #getEword# DESTINATION #gen(TO)#
246 //                             <verb> #getEword# #gen(ACTION)#
247 //                             <tense> #gen(TENSE)# PERIOD
248 // Done by: Leouel Guanzon, Marco Flores
249 void after_object()
250 {
251     if (!disable_tracing){
252         cout << "Processing <afterObject>" << endl;
253     }
254     switch(next_token())
255     {
256         case WORD2:
257             verb();
258             getEword();
259             gen("ACTION");
260             tense();
261             gen("TENSE");
262             match(PERIOD);
263             break;
264         case WORD1:
265             noun();
266             getEword();
267             match(DESTINATION);
268             gen("TO");
269             verb();
270             getEword();
271             gen("ACTION");
272             tense();
273             gen("TENSE");
274             match(PERIOD);
275             break;
276         case PRONOUN:
277             match(PRONOUN);
278             getEword();
279             match(DESTINATION);
280             gen("TO");
281             verb();
282             getEword();
283             gen("ACTION");
284             tense();
285             gen("TENSE");
286             match(PERIOD);
287             break;
288         default:
289             syntaxerror2("afterObject", saved_lexeme);
290             break;
291     }
```

219,1-4 55%

guanz004@empress:~...

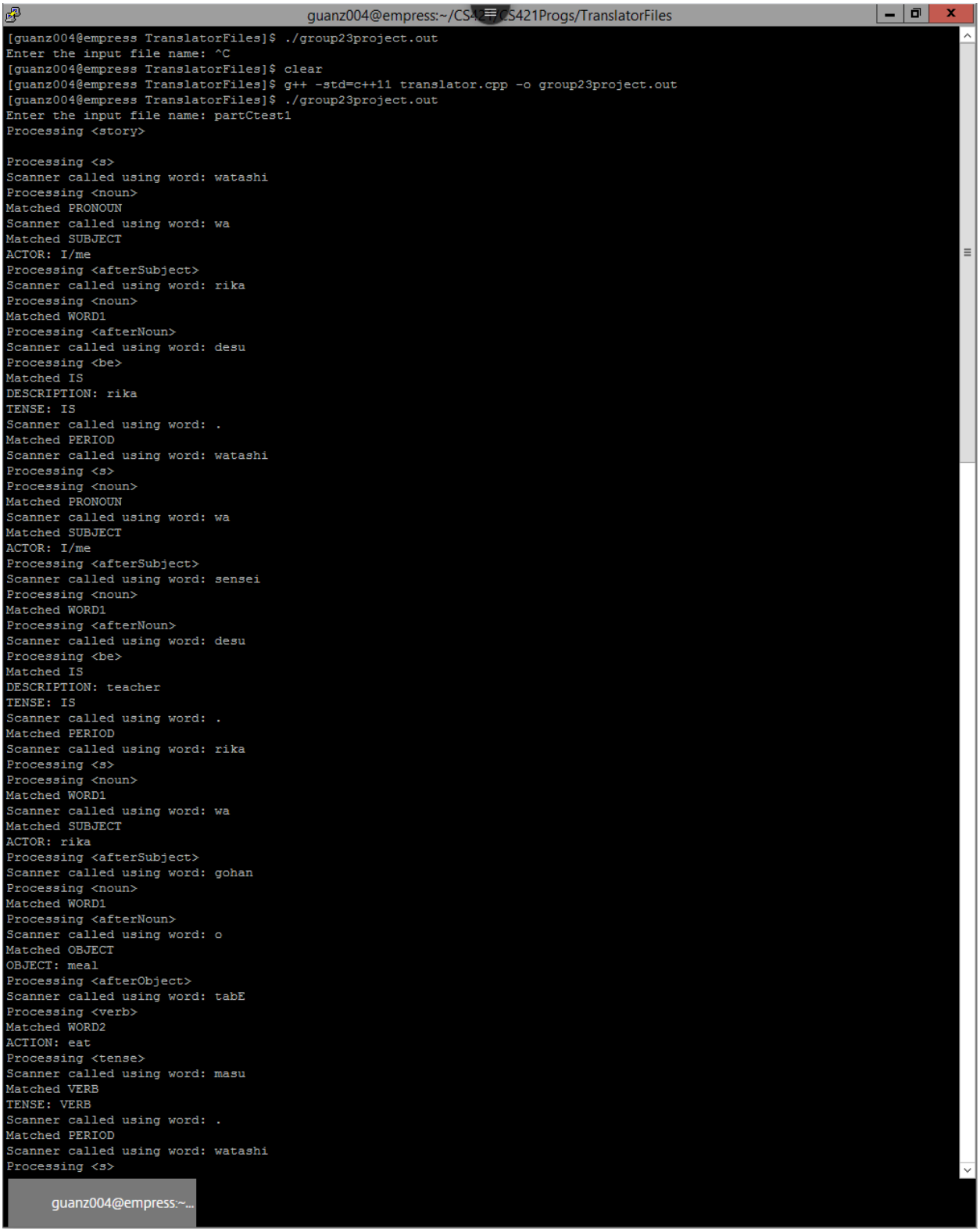

guanz004@empress:~/CS421Progs/TranslatorFiles
292 }
293
294 // Grammar: <after_noun> ::= <be> #gen(DESCRIPTION)# #gen(TENSE)# PERIOD |
295 //   DESTINATION #gen(TO)#
296 //   <verb> #getEword# #gen(ACTION)#
297 //   <tense> #gen(TENSE)# PERIOD |
298 //   OBJECT #gen(OBJECT)#
299 //   <after_object>
300 // Done by: Leouel Guanzon, Marco Flores
301 void after_noun()
302 {
303   if (!disable_tracing){
304   cout << "Processing <afterNoun>" << endl;
305   }
306   switch(next_token())
307   {
308   case IS:
309   be();
310   gen("DESCRIPTION");
311   gen("TENSE");
312   match(PERIOD);
313   break;
314   case WAS:
315   be();
316   gen("DESCRIPTION");
317   gen("TENSE");
318   match(PERIOD);
319   break;
320   case DESTINATION:
321   match(DESTINATION);
322   gen("TO");
323   verb();
324   getEword();
325   gen("ACTION");
326   tense();
327   gen("TENSE");
328   match(PERIOD);
329   break;
330   case OBJECT:
331   match(OBJECT);
332   gen("OBJECT");
333   after_object();
334   break;
335   default:
336   syntaxerror2("afterNoun", saved_lexeme);
337   break;
338   }
339 }
340
341 // Grammar: <after_subject> ::= <verb> #getEword# #gen(ACTION)#
342 //   <tense> #gen(TENSE)# PERIOD |
343 //   <noun> #getEword#
344 //   <aftern_noun>
345 // Done by: Leouel Guanzon, Marco Flores
346 void after_subject()
347 {
348   if (!disable_tracing){
349   cout << "Processing <afterSubject>" << endl;
350   }
351   switch(next_token())
352   {
353   case WORD2:
354   verb();
355   getEword();
356   gen("ACTION");
357   tense();
358   gen("TENSE");
359   match(PERIOD);
360   break;
361   case WORD1:
362   noun();
363   getEword();
364   after_noun();
364,1-4 74%
guanz004@empress:~...

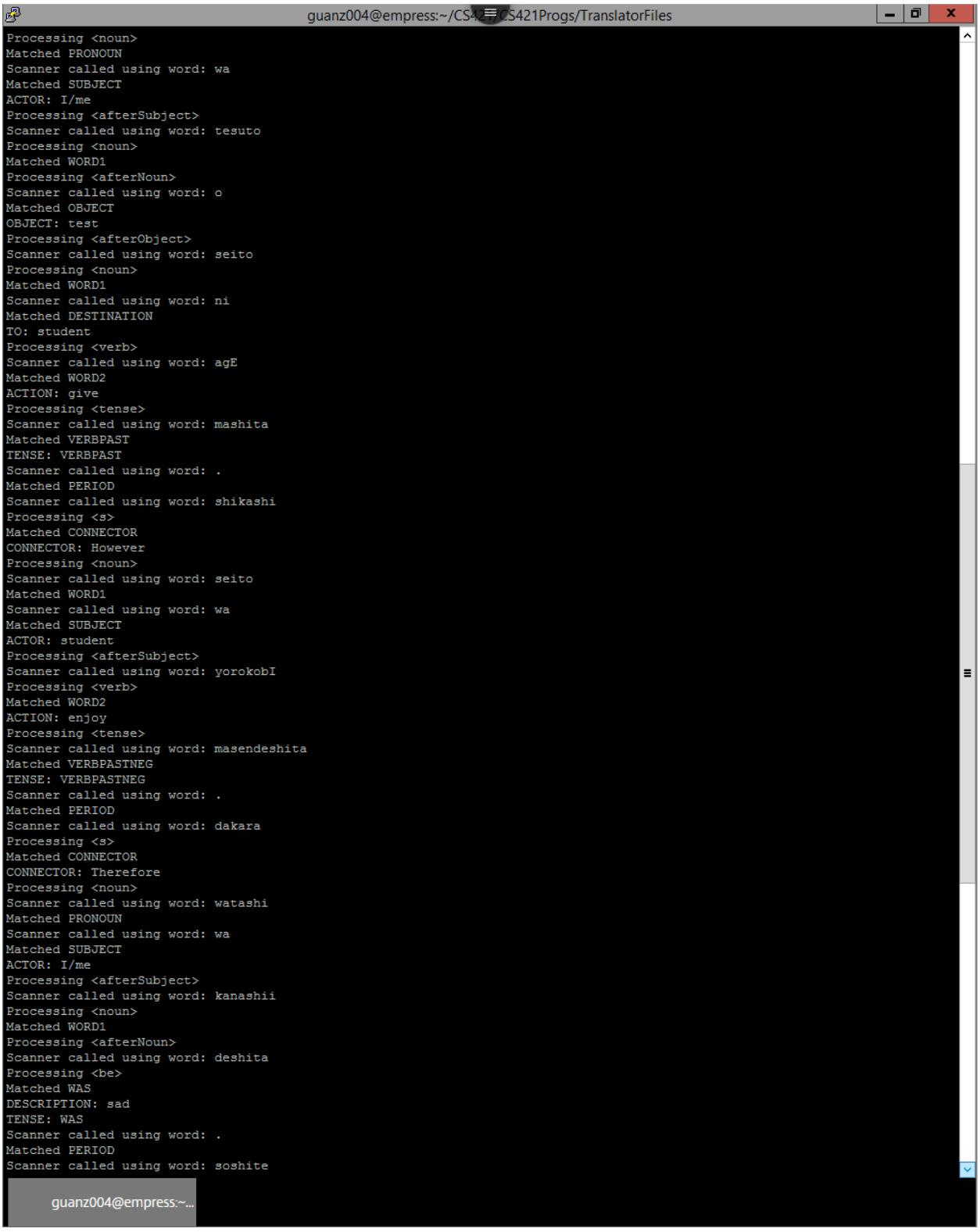

guanz004@empress:~/CS421Progs/TranslatorFiles
break;
case PRONOUN:
noun();
getEword();
after_noun();
break;
default:
syntaxerror2("afterSubject", saved_lexeme);
break;
}
}
// Grammar: <s> ::= [CONNECTOR #getEword# #gen(CONNECTOR)#]
//<noun> #getEword# SUBJECT #gen(ACTOR)#
//<after_subject>
// Done by: Leouel Guanzon, Marco Flores
void s()
{
if (!disable_tracing){
cout << "Processing <s>" << endl;
}
if(next_token() == CONNECTOR)
{
match(CONNECTOR);
getEword();
gen("CONNECTOR");
}
noun();
getEword();
match(SUBJECT);
gen("ACTOR");
after_subject();
}
// ---------- Driver ----------
// The final test driver to start the translator
// Done by: John Foster
int main(int argc, char* argv[])
{
// opens the lexicon.txt file and reads it into Lexicon
// closes lexicon.txt
string k,v;
fin.open("lexicon.txt");
if (!fin) {
cerr << "Can't open lexicon.txt";
return 1;
}
while (fin >> k >> v) {
lexicon_map[k] = v;
}
fin.close();
// opens the output file translated.txt
fout.open("translated.txt");
if (!fout) {
cerr << "Can't open translated.txt";
return 1;
}
// opens the output file translated.txt
ferr.open("errors.txt");
if (!ferr) {
cerr << "Can't open errors.txt";
return 1;
}
//** calls the <story> to start parsing
//** closes the input file
//** closes traslated.tx
437,27
92%
guanz004@empress:~...

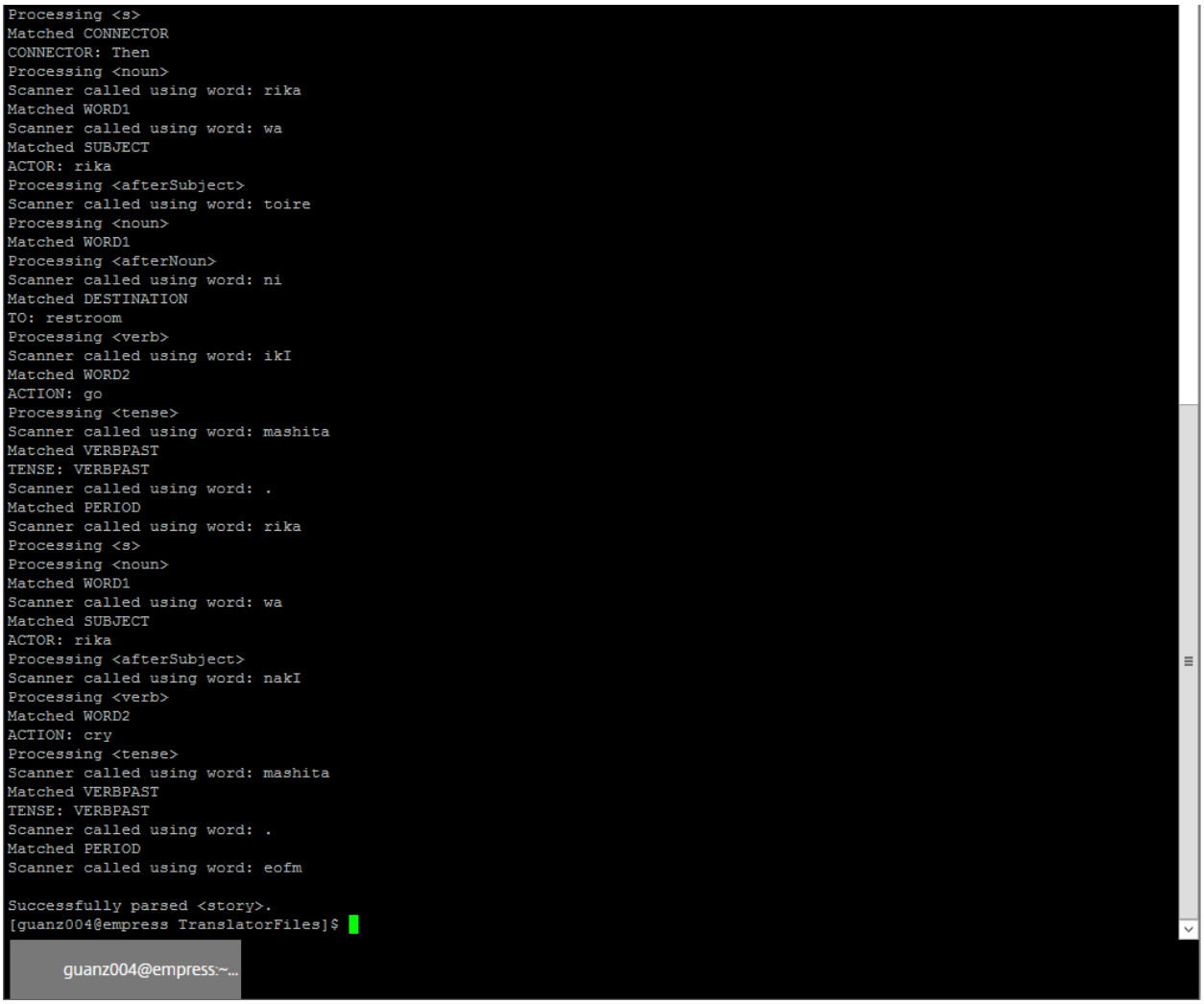


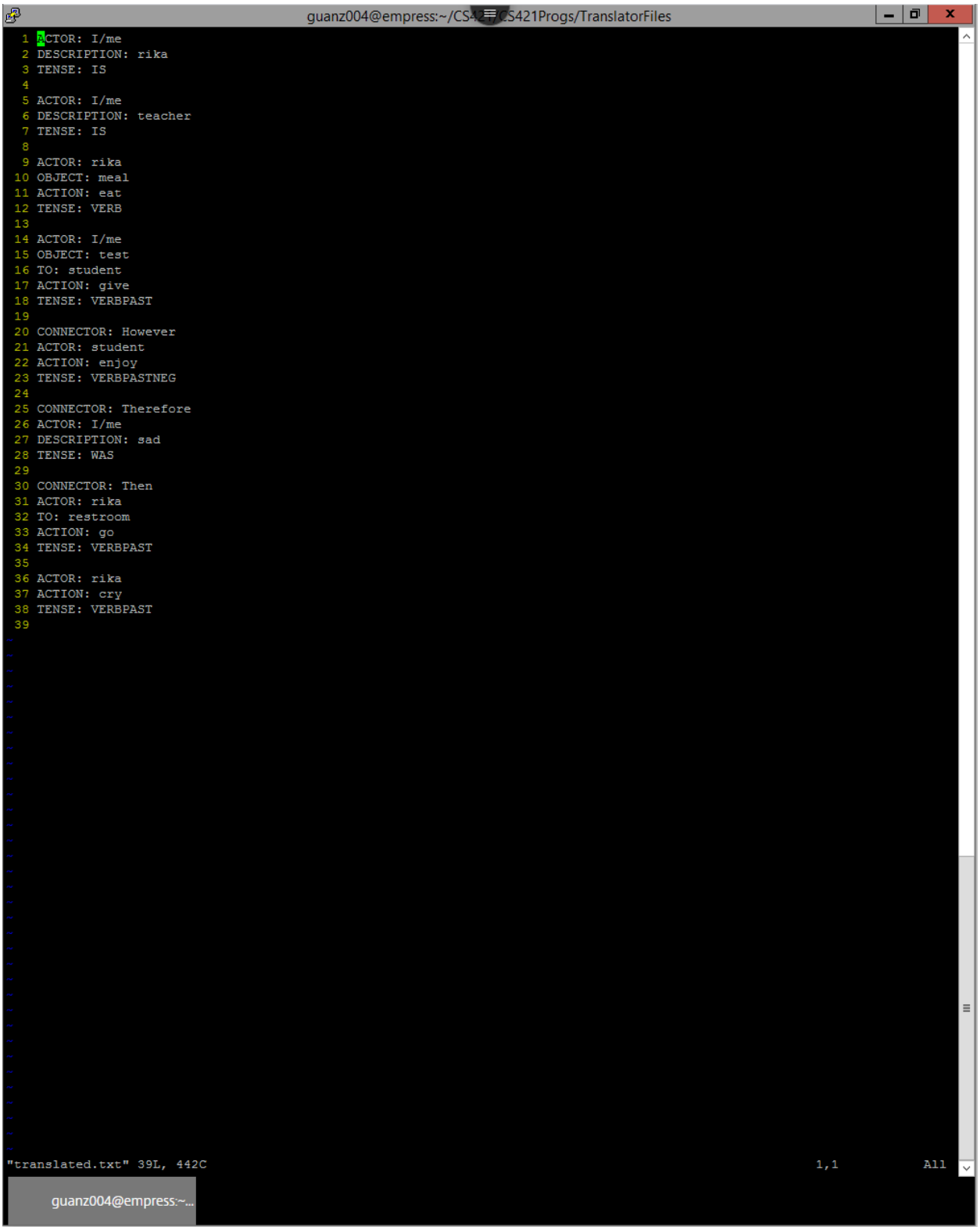
```
438   cout << "Enter the input file name: ";
439   cin >> filename;
440   fin.open(filename.c_str());
441
442   if (argc==2 && strcmp(argv[1],"-dissable") == 0){
443       disable_tracing = true;
444   }
445   if (!disable_tracing){
446       cout << "Processing <story>\n" << endl;
447   }
448   // Grammar: <story> ::= <s> { <s> }
449   s();
450   fout << endl;
451
452   while(next_token() != EOFM)
453   {
454       s();
455       fout << endl;
456   }
457
458   fin.close();
459   fout.close();
460
461 }// end
462 //** require no other input files!
463 //** syntax error EC requires producing errors.txt of error messages
464 //** tracing On/Off EC requires sending a flag to trace message output functions
465
                                                                  465,0-1       Bot
```

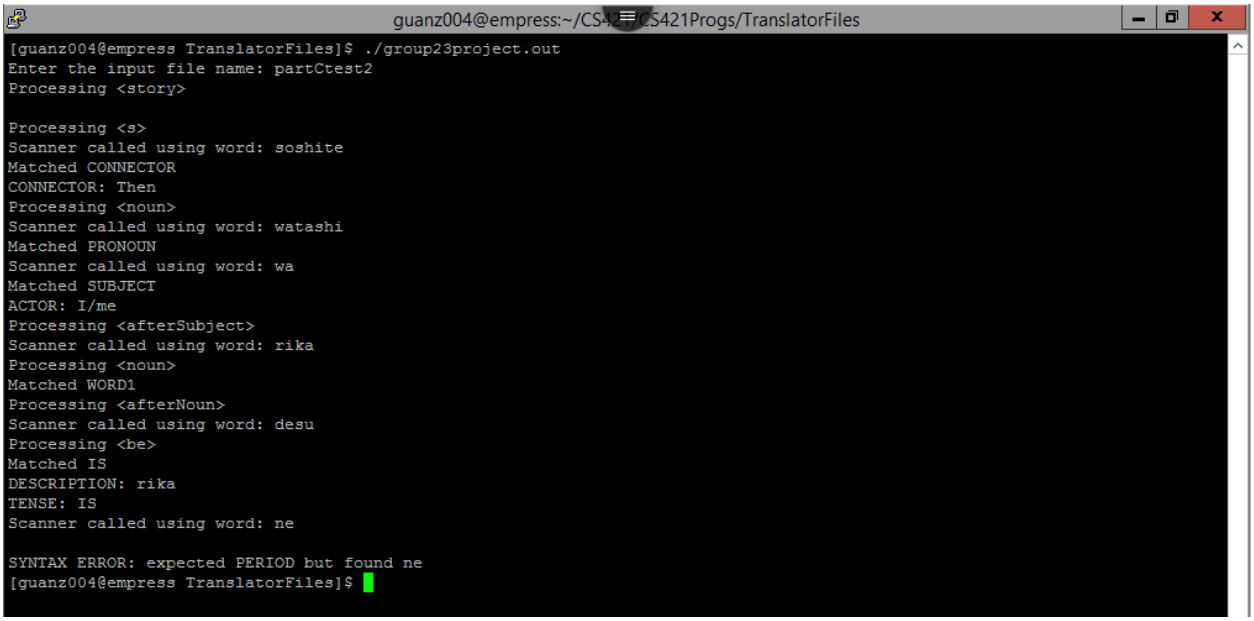
6 – Final test results



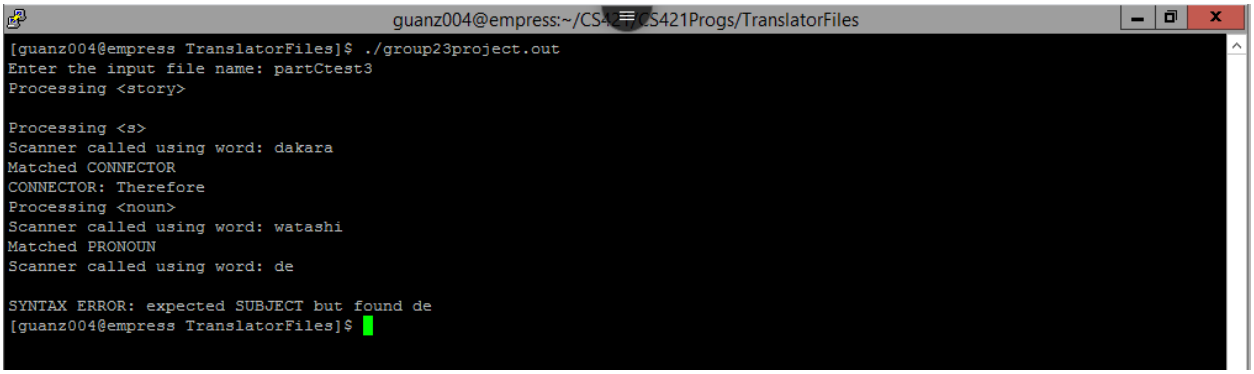

guanz004@empress:~/CS421Progs/TranslatorFiles
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
ACTOR: I/me
Processing <afterSubject>
Scanner called using word: tesuto
Processing <noun>
Matched WORD1
Processing <afterNoun>
Scanner called using word: o
Matched OBJECT
OBJECT: test
Processing <afterObject>
Scanner called using word: seito
Processing <noun>
Matched WORD1
Scanner called using word: ni
Matched DESTINATION
TO: student
Processing <verb>
Scanner called using word: agE
Matched WORD2
ACTION: give
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
TENSE: VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: shikashi
Processing <s>
Matched CONNECTOR
CONNECTOR: However
Processing <noun>
Scanner called using word: seito
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
ACTOR: student
Processing <afterSubject>
Scanner called using word: yorokobI
Processing <verb>
Matched WORD2
ACTION: enjoy
Processing <tense>
Scanner called using word: masendeshita
Matched VERBPASTNEG
TENSE: VERBPASTNEG
Scanner called using word: .
Matched PERIOD
Scanner called using word: dakara
Processing <s>
Matched CONNECTOR
CONNECTOR: Therefore
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
ACTOR: I/me
Processing <afterSubject>
Scanner called using word: kanashii
Processing <noun>
Matched WORD1
Processing <afterNoun>
Scanner called using word: deshita
Processing <be>
Matched WAS
DESCRIPTION: sad
TENSE: WAS
Scanner called using word: .
Matched PERIOD
Scanner called using word: soshite
guanz004@empress:~...

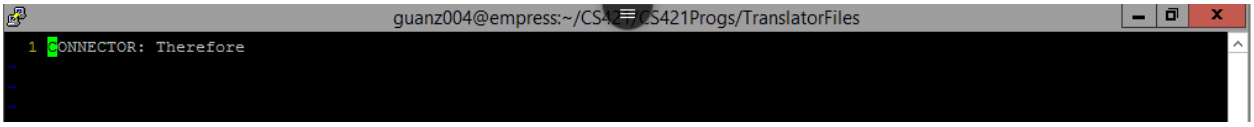

Processing <s>
Matched CONNECTOR
CONNECTOR: Then
Processing <noun>
Scanner called using word: rika
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
ACTOR: rika
Processing <afterSubject>
Scanner called using word: toire
Processing <noun>
Matched WORD1
Processing <afterNoun>
Scanner called using word: ni
Matched DESTINATION
TO: restroom
Processing <verb>
Scanner called using word: ikI
Matched WORD2
ACTION: go
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
TENSE: VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: rika
Processing <s>
Processing <noun>
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
ACTOR: rika
Processing <afterSubject>
Scanner called using word: nakI
Processing <verb>
Matched WORD2
ACTION: cry
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
TENSE: VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: eofm
Successfully parsed <story>.
[guanz004@empress TranslatorFiles]$
guanz004@empress:~...

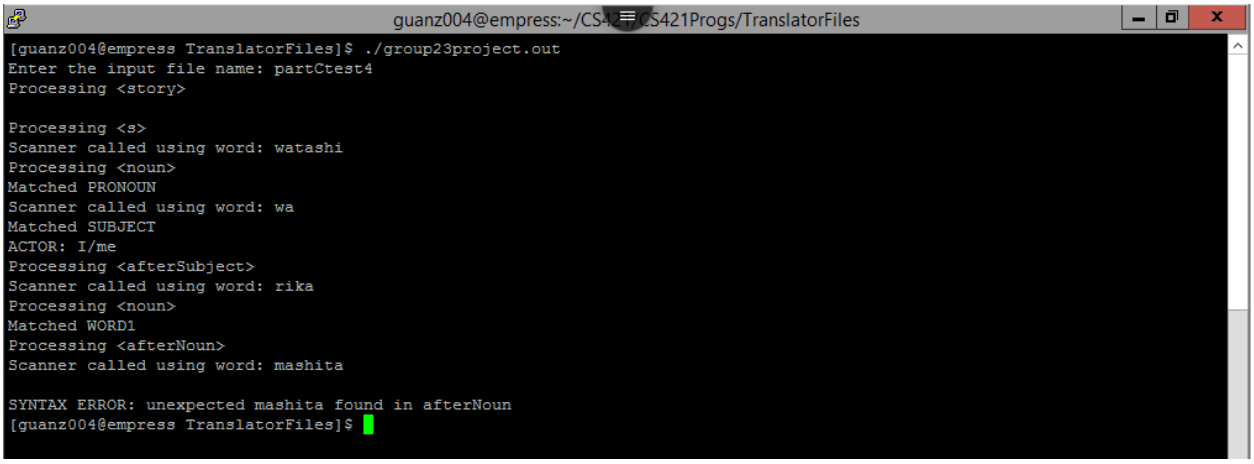

guanz004@empress:~/CS421Progs/TranslatorFiles
ACTOR: I/me
DESCRIPTION: rika
TENSE: IS
ACTOR: I/me
DESCRIPTION: teacher
TENSE: IS
ACTOR: rika
OBJECT: meal
ACTION: eat
TENSE: VERB
ACTOR: I/me
OBJECT: test
TO: student
ACTION: give
TENSE: VERBPAST
CONNECTOR: However
ACTOR: student
ACTION: enjoy
TENSE: VERBPASTNEG
CONNECTOR: Therefore
ACTOR: I/me
DESCRIPTION: sad
TENSE: WAS
CONNECTOR: Then
ACTOR: rika
TO: restroom
ACTION: go
TENSE: VERBPAST
ACTOR: rika
ACTION: cry
TENSE: VERBPAST
"translated.txt" 39L, 442C
1,1
All
guanz004@empress:~...

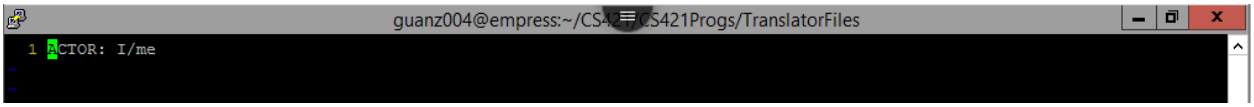

guanz004@empress:~/CS4
CS421Progs/TranslatorFiles
[guanz004@empress TranslatorFiles]$ ./group23project.out
Enter the input file name: partCtest2
Processing <story>
Processing <s>
Scanner called using word: soshite
Matched CONNECTOR
CONNECTOR: Then
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
ACTOR: I/me
Processing <afterSubject>
Scanner called using word: rika
Processing <noun>
Matched WORD1
Processing <afterNoun>
Scanner called using word: desu
Processing <be>
Matched IS
DESCRIPTION: rika
TENSE: IS
Scanner called using word: ne
SYNTAX ERROR: expected PERIOD but found ne
[guanz004@empress TranslatorFiles]$


guanz004@empress:~/CS4
CS421Progs/TranslatorFiles
1 CONNECTOR: Then
2 ACTOR: I/me
3 DESCRIPTION: rika
4 TENSE: IS

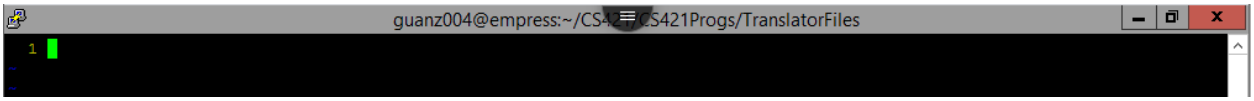

guanz004@empress:~/CS4
CS421Progs/TranslatorFiles
[guanz004@empress TranslatorFiles]$ ./group23project.out
Enter the input file name: partCtest3
Processing <story>
Processing <s>
Scanner called using word: dakara
Matched CONNECTOR
CONNECTOR: Therefore
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: de
SYNTAX ERROR: expected SUBJECT but found de
[guanz004@empress TranslatorFiles]$


guanz004@empress:~/CS4
CS421Progs/TranslatorFiles
1 CONNECTOR: Therefore

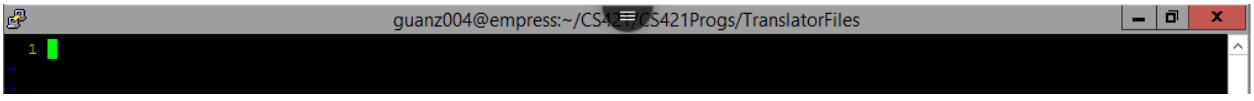

guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
[guanz004@empress TranslatorFiles]$ ./group23project.out
Enter the input file name: partCtest4
Processing <story>
Processing <s>
Scanner called using word: watashi
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
ACTOR: I/me
Processing <afterSubject>
Scanner called using word: rika
Processing <noun>
Matched WORD1
Processing <afterNoun>
Scanner called using word: mashita
SYNTAX ERROR: unexpected mashita found in afterNoun
[guanz004@empress TranslatorFiles]$


guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
1 ACTOR: I/me


guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
[guanz004@empress TranslatorFiles]$ ./group23project.out
Enter the input file name: partCtest5
Processing <story>
Processing <s>
Scanner called using word: wa
Processing <noun>
SYNTAX ERROR: unexpected wa found in noun
[guanz004@empress TranslatorFiles]$


guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
1


guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
[guanz004@empress TranslatorFiles]$ ./group23project.out
Enter the input file name: partCtest6
Processing <story>
Processing <s>
Lexical error: apple is not a valid token.
Scanner called using word: apple
Processing <noun>
SYNTAX ERROR: unexpected apple found in noun
[guanz004@empress TranslatorFiles]$


guanz004@empress:~/CS421/CS421Progs/TranslatorFiles
1