0- State of the program statement

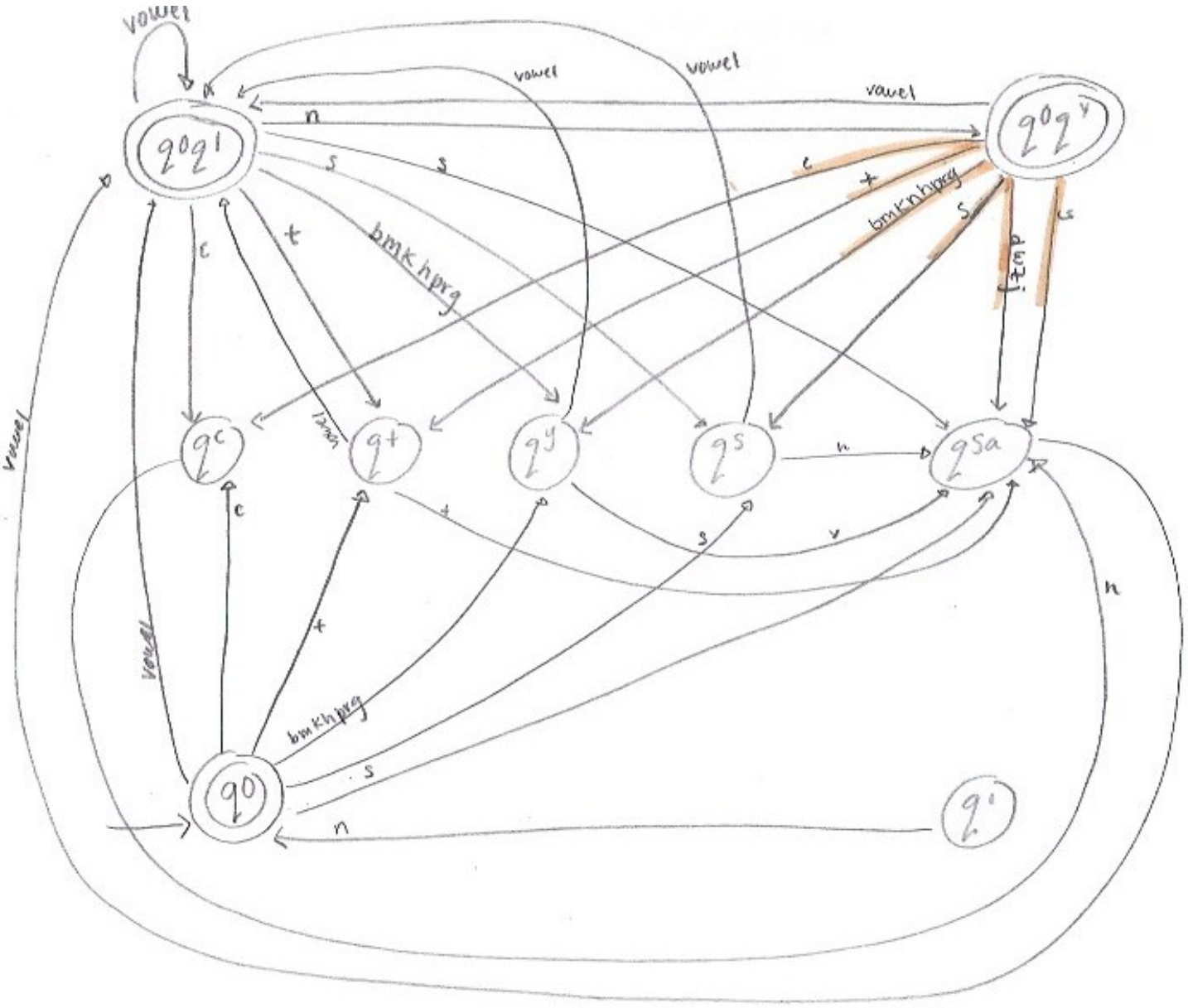
CS421 Project – Japanese Project
Group #15 –
Ronalyn Castilla, Tyler Cochran, Anjelina Velazquez

- Working perfectly? Yes

- List of parts you did not complete? Yes
  List of bugs? NA

- Descriptions of extra credit features? None

1 - DFA



2- Scanner code

```
#include<iostream>
#include<fstream>
#include<string>
#include <map>
using namespace std;

/* Look for all **'s and complete them */

//=====================================================
// File scanner.cpp written by: Group Number: **15
//Anjelina, Ronalyn, and Tyler
//=====================================================

// --------- Two DFAs ---------------------------------

// WORD DFA
// Done by: anjelina v
// RE: (vowel | vowel n |bmknhprg vowel| bmknhprg vowel n| dwzyj
vowel| dwzyj vowel n| bmknhprg y vowel| bmknhprg y vowel n| t vowel| t
vowel n| t s vowel| t s vowel n| s vowel| s vowel n|s h vowel| s h
vowel n| ch vowel| ch vowel n) ^+
bool word (string s)
{

  int state = 0;
  int charpos = 0;
  /* replace the following todo the word dfa  ** Prof. Krell notes
given
  while (s[charpos] != '\0')
    {
      if (state == 0 && s[charpos] == 'a')
      state = 1;
      else
      if (state == 1 && s[charpos] == 'b')
      state = 2;
      else
      if (state == 2 && s[charpos] == 'b')
      state = 2;
      else
           return(false);
      charpos++;
    }//end of while

  // where did I end up????
  if (state == 2) return(true);  // end in a final state
   else return(false);
  */

  /* why I used case of if... both do them do the same thig I find
switch statements allow
```

```
for cleaner/ nicer code. Swicth statements to me seem easier to use
especially when comparing
multiple experssions like in this funcation.
Ronalyn, when checking this if you see anything wrong with my switch
statemens let me know. THANK YOU!
   */
  while (s[charpos] != '\0')
    {   //this is the start of q0
      if (state == 0){
        switch(s[charpos]) {
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        case 'd': case 'w': case 'z': case 'y': case 'j':
          state = 2; break;
        case 'b': case 'm': case 'k': case 'n': case 'h': case 'p':
case 'r': case 'g':
          state = 6; break;
        case 't':
          state = 3; break;
        case 's':
          state = 4; break;
        case 'c':
          state = 5;  break;
        default:
          return false;
        }
      } // end q0

      // start switch of q0 q1
      else if (state == 1) {
        switch(s[charpos]) {
        case 'n':
          state = 0; break;
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        case 'd': case 'w': case 'z': case 'y': case 'j':
          state = 2; break;
        case 't':
          state = 3; break;
        case 's':
          state = 4; break;
        case 'c':
          state = 5;  break;
        case 'b': case 'm': case 'k': case 'h': case 'p': case 'r':
case 'g':
          state = 6; break;
        default:
          return false;
```

```
        }
      } // end switch q0 q1

      //(small) start switch qsa
      else if (state == 2) {
        switch(s[charpos]) {
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        default:
          return false;
        }
      }//end of qsa switch

      // start of qt
      else  if (state == 3) {
        switch(s[charpos]) {
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        case 's':
          state = 4; break;
        default:
          return false;
        }
      } // end of qt switch

        //START of qs
      else if (state == 4) {
        switch(s[charpos]) {
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        case 'h':
          state = 2; break;
        default:
          return false;
        }
      }// end of qs switch

      //start of qc
      else if (state == 5) {
        switch(s[charpos]) {
        case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
          state = 1; break;
        case 'h':
          state = 2; break;
        default:
          return false;
```

```
        }
      } // end of switch qc

        //START OF qr
      else if (state == 6) {
         switch(s[charpos])
           {
           case 'a': case 'e': case 'i': case 'o': case 'u': case 'I':
case 'E':
             state = 1; break;
           case 'y' : state = 2; break;
           default:
             return false;
           }
      } // end of qr

      else {
         cout <<"STUCK"<<endl;
         return false;
      }
      charpos++;
    }//end of while

  // where did I end up????
  if (state == 0 || state == 1) return(true);  // end in a final state
  else return(false);
}


// PERIOD DFA
// Done by: ANJLEINA V
bool period (string s)
{
  // complete this **
  if(s[0] == '.') return true;
  else return false;
}

// ------ Three  Tables ------------------------------------

// TABLES Done by: anjelina V

// ** Update the tokentype to be WORD1, WORD2, PERIOD, ERROR, EOFM,
etc.
//Ronalyn let me know if I missed anything I'vebeen looking at this
for
//too long. I feel like it should be correct.
enum tokentype {WORD1, WORD2, PERIOD, VERB, VERBNEG, VERBPAST,
VERBPASTNEG, IS, WAS, OBJECT, SUBJECT, DESTINATION, PRONOUN,
CONNECTOR, ERROR, EOFM};
```

```
// ** For the display names of tokens - must be in the same order as
the tokentype.
string tokenName[30] = {"WORD1", "WORD2", "PERIOD", "VERB", "VERBNEG",
"VERBPAST", "VERBPASTNEG", "IS", "WAS", "OBJECT", "SUBJECT",
"DESTINATION", "PRONOUN", "CONNECTOR", "ERROR", "EOFM"};

// ** Need the reservedwords table to be set up here.
// ** Do not require any file input for this. Hard code the table.
// ** a.out should work without any additional files.
//Hey Ronalyn let me know how you feel about what I did below.
//It is funcationing correcetly but I've been looking at this for so
//long everything looks right

map<string,tokentype> reservedWords;
map<string,tokentype> :: iterator it;

void createMap(){
  reservedWords["masu"] = VERB;
  reservedWords["masen"] = VERBNEG;
  reservedWords["mashita"] = VERBPAST;
  reservedWords["masendeshita"] = VERBPASTNEG;
  reservedWords["desu"] = IS;
  reservedWords["deshita"] = WAS;
  reservedWords["o"] = OBJECT;
  reservedWords["wa"]=SUBJECT;
  reservedWords["ni"] = DESTINATION;
  reservedWords["watashi"] = PRONOUN;
  reservedWords["anata"]= PRONOUN;
  reservedWords["kare"]=PRONOUN;
  reservedWords["kanojo"]=PRONOUN;
  reservedWords["sore"]=PRONOUN;
  reservedWords["mata"]=CONNECTOR;
  reservedWords["soshite"]=CONNECTOR;
  reservedWords["shikashi"]=CONNECTOR;
  reservedWords["dakara"]=CONNECTOR;
  reservedWords["eofm"]=EOFM;
}

// ------------ Scanner and Driver ----------------------

ifstream fin;  // global stream for reading from the input file
string nextWord;
// Scanner processes only one word each time it is called
// Gives back the token type and the word itself

// ** Done by:ANJELIna V
int scanner(tokentype& tt, string& w)
{
  // ** Grab the next word from the file via fin
```

```
  // 1. If it is eofm, return right now.

  /*  **
  2. Call the token functions (word and period)
     one after another (if-then-else).
     Generate a lexical error message if both DFAs failed.
     Let the tokentype be ERROR in that case.

  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)
  */
    fin >> w;
    if(w == "eofm") return-1;

    if(word(w)) {
      it = reservedWords.find(w);

      if(it != reservedWords.end()){
        tt = it->second;
      }
      else if(w[w.length()-1] == 'I' || w[w.length()-1] == 'E') {
        tt = WORD2;
      }
      else {
        tt = WORD1;
      }
    }else if(period(w)) {
      tt =  PERIOD;
    } else {
      tt = ERROR;
      cout << "Lexical error: " << w << " is not a valid token"
<<endl;
    }
    return 0;
}//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()
{
  createMap();
  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)
    {
       scanner(thetype, theword);  // call the scanner which sets
                                   // the arguments
       if (theword == "eofm") break;  // stop now

       cout << "Type is:" << tokenName[thetype] << endl;
       cout << "Word is:" << theword << endl;
    }

   cout << "End of file is encountered." << endl;
   fin.close();

}// end
```

3- Scanner test results - Test1

```
Script started on Tue 15 Dec 2020 04:47:19 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles      [?1034h[cowie001@empress ScannerFiles]$
g++ scanner.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles   [cowie001@empress ScannerFiles]$ ./a.out
Enter the input file name: scannertest1
Type is:PRONOUN
Word is:watashi
Type is:SUBJECT
Word is:wa
Type is:WORD1
Word is:rika
Type is:IS
Word is:desu
Type is:PERIOD
Word is:.
Type is:PRONOUN
Word is:watashi
Type is:SUBJECT
Word is:wa
Type is:WORD1
Word is:sensei
Type is:IS
Word is:desu
Type is:PERIOD
Word is:.
Type is:PRONOUN
Word is:watashi
Type is:SUBJECT
Word is:wa
Type is:WORD1
Word is:ryouri
Type is:OBJECT
Word is:o
Type is:WORD2
Word is:yarI
Type is:VERB
Word is:masu
Type is:PERIOD
Word is:.
Type is:PRONOUN
Word is:watashi
Type is:SUBJECT
Word is:wa
Type is:WORD1
Word is:gohan
Type is:OBJECT
Word is:o
Type is:WORD1
Word is:seito
Type is:DESTINATION
Word is:ni
Type is:WORD2
Word is:agE
```

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
Word is:mashita
Type is:PERIOD
Word is:.

End of file is encountered.
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles [cowie001@empress ScannerFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:47:37 PM PST

3- Scanner test results- Test2

```
Script started on Tue 15 Dec 2020 04:47:46 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles      [?1034h[cowie001@empress ScannerFiles]$
g++ scanner.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles   [cowie001@empress ScannerFiles]$ ./a.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
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

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
Type is:WORD1
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

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.
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/ScannerFiles [cowie001@empress ScannerFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:48:10 PM PST

4- Factored rules with new non-terminal names and semantic routines

<s> ::= [CONNECTOR] <noun> SUBJECT <aftersubject>

<aftersubject> ::= <verb><tense> PERIOD } <noun> <afternoun>

<afternoun> ::= <be> PERIOD | DESTINATION <verb><tense> PERIOD | OBJECT< afterobject>

<afterobject>::= <verb> <tense> PERIOD | <noun> DESTINATION <verb> <tense> PERIOD

PART C – GRAMMER RULES

<story> ::= <s> {<s>}

<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

<noun> ::= WORD1 | PRONOUN
<verb> ::= WORD 2
<be> ::= IS | WAS
<tense> ::= VERBPAST | VERBPASTENG | VERB | VERBENG

5- Parser/Translator Code

```
#include<iostream>
#include<fstream>
#include<string>
#include "scanner.cpp"
#include <stdlib.h>
#include <vector>
using namespace std;

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

//=================================================
// File translator.cpp written by Group Number: 15
//=================================================

void sNonterm();
void afterSubjectNonterm();
void afterNounNonterm();
void afterObjectNonterm();
void nounNonterm();
void verbNonterm();
void beNonterm();
void tenseNonterm();

vector <string> wordJ;
vector <string> wordE;

string savedEword;

ofstream outFile;

string saved_lexeme;
tokentype saved_token;
bool token_available = false;

// ----- 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: Tyler Cochran


// ** 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: Tyler Cochran

void getEword()
{
  bool found = false;
  for(int a = 0; a < wordJ.size(); a++)
    {
      if(wordJ[a] == saved_lexeme)
        {
          savedEword = wordE[a];
          found = true;
        }
    }
  if(found == false)
    {
      savedEword = saved_lexeme;
    }
}

//    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: Tyler Cochran

void gen(string line_type)
{
  if(line_type == "TENSE")
    {
      outFile << line_type << " " << tokenName[saved_token] << endl;
    }
  else
    {
      outFile << line_type << " " << savedEword << endl;
    }
}

// ----- Changes to the parser.cpp content --------------------

// ** Comment update: Be sure to put the corresponding grammar
//    rule with semantic routine calls
//    above each non-terminal function

// ** Each non-terminal function should be calling
//    getEword and/or gen now.

//================================================
// File parser.cpp written by Group Number: **15
//================================================
```

```
// ----- Four Utility Functions and Globals
-----------------------------------

// ** Need syntaxerror1 and syntaxerror2 functions (each takes 2 args)
//    to display syntax error messages as specified by me.


// Type of error: **Matching error
// Done by: **Ronalyn Castilla
void syntaxerror1(string lexeme, tokentype expected) {
  cout << "\nSYNTAX ERROR: expected " << tokenName[expected] << " but
found " << lexeme << "\n";
  exit(1);
}
// Type of error: **Error in parser
// Done by: **Ronalyn Castilla
void syntaxerror2(string lexeme, string parserFunction) {
  cout << "\nSYNTAX ERROR: unexpected " << lexeme << " found in " <<
parserFunction << "\n";
  exit(1);
}

// ** Need the updated match and next_token with 2 global vars
// saved_token and saved_lexeme

// Purpose: **Save the token, the lexeme, and the flag if a token is
available. Set the flag if there is a token available
// Done by: **Ronalyn Castilla
tokentype next_token() {
  if(!token_available) {
    //If token_available is false then eat a token on the scanner and
save it
    scanner(saved_token, saved_lexeme);
    token_available = true;
  }
  //Return the saved token
  return saved_token;
}

// Purpose: **To match a given token type with the saved token
// Done by: **Ronalyn Castilla
bool match(tokentype expected) {
  //Check if there is not a match
  if(next_token() != expected)
    {
      //Matching error
      syntaxerror1(saved_lexeme, expected);
    }
  else
```

```
    {
      //If there is a match, eat the token on the scanner and print
that there was a match
      token_available = false;
      cout << "Matched " << tokenName[expected] << "\n";
      return true;
    }
}

// ----- RDP functions - one per non-term ------------------

// ** Make each non-terminal into a function here
// ** Be sure to put the corresponding grammar rule above each
function
// ** Be sure to put the name of the programmer above each function

// Grammar: **<noun> ::= WORD1 | PRONOUN
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void nounNonterm() {
  cout << "Processing <noun>\n";
  switch(next_token())
    {
    case WORD1:
      match(WORD1);
      getEword();
      break;
    case PRONOUN:
      match(PRONOUN);
      getEword();
      break;
    default:
      syntaxerror2(saved_lexeme, "noun");
    }
}

// Grammar: **<verb> ::= WORD2
// Done by: **Ronalyn Castilla
// Updated by Tyler Cochran
void verbNonterm() {
  cout << "Processing <verb>\n";
  switch(next_token())
    {
    case WORD2:
      match(WORD2);
      getEword();
      gen("ACTION");
      break;
    default:
      syntaxerror2(saved_lexeme, "verb");
```

```
    }
}

// Grammar: **<be> ::= IS | WAS
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void beNonterm() {
  cout << "Processing <be>\n";
  gen("DESCRIPTION");
  switch(next_token())
    {
    case IS:
      match(IS);
      gen("TENSE");
      break;
    case WAS:
      match(WAS);
      gen("TENSE");
      break;
    default:
      syntaxerror2(saved_lexeme, "be");
    }
}

// Grammar: **<tense> ::= VERBPAST | VERBPASTNEG | VERB | VERBNEG
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void tenseNonterm() {
  cout << "Processing <tense>\n";
  switch(next_token())
    {
    case VERBPAST:
      match(VERBPAST);
      gen("TENSE");
      break;
    case VERBPASTNEG:
      match(VERBPASTNEG);
      gen("TENSE");
      break;
    case VERB:
      match(VERB);
      gen("TENSE");
      break;
    case VERBNEG:
      match(VERBNEG);
      gen("TENSE");
      break;
    default:
      syntaxerror2(saved_lexeme, "tense");
    }
```

```
}

// Grammar: **<afterObject> ::= <verb> #getEword# #gen("ACTION")#
<tense> #gen("TENSE")# PERIOD | <noun> #getEword# DESTINATION
#gen("TO")# <verb> #getEword# #gen("ACTION")# <tense> #gen("TENSE")#
PERIOD
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void afterObjectNonterm() {
  cout << "Processing <afterObject>\n";
  switch(next_token())
    {
    case WORD2:
      verbNonterm();
      tenseNonterm();
      match(PERIOD);
      break;
    case WORD1:
      nounNonterm();
      match(DESTINATION);
      gen("TO");
      verbNonterm();
      tenseNonterm();
      match(PERIOD);
      break;
    case PRONOUN:
      nounNonterm();
      match(DESTINATION);
      gen("TO");
      verbNonterm();
      tenseNonterm();
      match(PERIOD);
      break;
    default:
      syntaxerror2(saved_lexeme, "afterObject");
    }
}

// Grammar: **<afterNoun> ::= <be> #gen("DESCRIPTION")# #gen("TENSE")#
PERIOD | DESTINATION #gen("TO")# <verb> #getEword# #gen("ACTION")
<tense> #gen("TENSE")# PERIOD | OBJECT #gen("OBJECT")# <afterObject>
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void afterNounNonterm() {
  cout << "Processing <afterNoun>\n";
  switch(next_token())
    {
    case IS:
      beNonterm();
      match(PERIOD);
```

```
      break;
    case WAS:
      beNonterm();
      match(PERIOD);
      break;
    case DESTINATION:
      match(DESTINATION);
      gen("TO");
      verbNonterm();
      tenseNonterm();
      match(PERIOD);
      break;
    case OBJECT:
      match(OBJECT);
      gen("OBJECT");
      afterObjectNonterm();
      break;
    default:
      syntaxerror2(saved_lexeme, "afterNoun");
    }
}

// Grammar: **<afterSubject> ::= <verb> #getEword# #gen("ACTION")#
<tense> #gen("TENSE")# PERIOD | <noun> #getEword# <afterNoun>
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void afterSubjectNonterm() {
  cout << "Processing <afterSubject>\n";
  switch(next_token())
    {
    case WORD2:
      verbNonterm();
      tenseNonterm();
      match(PERIOD);
      break;
    case WORD1:
      nounNonterm();
      afterNounNonterm();
      break;
    case PRONOUN:
      nounNonterm();
      afterNounNonterm();
      break;
    default:
      syntaxerror2(saved_lexeme, "afterSubject");
    }
}

// Grammar: **<s> ::= [CONNECTOR #getEword# #gen("CONNECTOR")] <noun>
#getEword# SUBJECT gen("ACTOR") <afterSubject>
```

```
// Done by: **Ronalyn Castilla
// Updated by: Tyler Cochran
void sNonterm() {
  next_token();
  if(saved_lexeme != "eofm")
    {
      cout << "Processing <s>\n";

      if(next_token() == CONNECTOR) {
        match(CONNECTOR);
        getEword();
        gen("CONNECTOR");
      }

      nounNonterm();
      match(SUBJECT);
      gen("ACTOR");
      afterSubjectNonterm();
    }
}

// Grammar: **<story> ::= <s> {<s>}
// Done by: **Ronalyn Castilla
void storyNonterm() {
  cout << "Processing <story>\n\n";
  sNonterm();
  while(true && (saved_lexeme != "eofm"))
    {
      outFile << endl;
      sNonterm();
    }
  cout << "\nSuccessfully parsed <story>.\n";
}

// ---------------- Driver ----------------------------

// The final test driver to start the translator
// Done by:  Tyler Cochran
int main()
{
  //** opens the lexicon.txt file and reads it into Lexicon
  ifstream input;
  string tJ;
  string tE;
  input.open("lexicon.txt");
  cout << "Opening File" << endl;

  while(input)
    {
      input >> tJ;
```

```
      input >> tE;
      wordJ.push_back(tJ);
      wordE.push_back(tE);
    }

  //** closes lexicon.txt
  input.close();

  //** opens the output file translated.txt
  outFile.open("translated.txt");

  string filename;
  cout << "Enter the input file name: ";
  cin >> filename;
  fin.open(filename.c_str());

  createMap();

  //** calls the <story> to start parsing
  storyNonterm();

  //** closes the input file
  fin.close();

  //** closes traslated.txt
  outFile.close();
}// end

//** require no other input files!
//** syntax error EC requires producing errors.txt of error messages
//** tracing On/Off EC requires sending a flag to trace message output
functions
```

6- Test results - Test1

```
Script started on Tue 15 Dec 2020 04:49:49 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ translator.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest1
Processing <story>

Processing <s>
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Processing <be>
Matched IS
Matched PERIOD
Processing <s>
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Processing <be>
Matched IS
Matched PERIOD
Processing <s>
Processing <noun>
Matched WORD1
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Matched OBJECT
Processing <afterObject>
Processing <verb>
Matched WORD2
Processing <tense>
Matched VERB
Matched PERIOD
Processing <s>
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
```

Matched OBJECT
Processing <afterObject>
Processing <noun>
Matched WORD1
Matched DESTINATION
Processing <verb>
Matched WORD2
Processing <tense>
Matched VERBPAST
Matched PERIOD
Processing <s>
Matched CONNECTOR
Processing <noun>
Matched WORD1
Matched SUBJECT
Processing <afterSubject>
Processing <verb>
Matched WORD2
Processing <tense>
Matched VERBPASTNEG
Matched PERIOD
Processing <s>
Matched CONNECTOR
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Processing <be>
Matched WAS
Matched PERIOD
Processing <s>
Matched CONNECTOR
Processing <noun>
Matched WORD1
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Matched DESTINATION
Processing <verb>
Matched WORD2
Processing <tense>
Matched VERBPAST
Matched PERIOD
Processing <s>
Processing <noun>
Matched WORD1
Matched SUBJECT
Processing <afterSubject>
Processing <verb>

Matched WORD2
Processing <tense>
Matched VERBPAST
Matched PERIOD

Successfully parsed <story>.
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:50:17 PM PST

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

6- Test results - Test2

```
Script started on Tue 15 Dec 2020 04:50:22 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ tran   slator.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest2
Processing <story>

Processing <s>
Matched CONNECTOR
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>
Processing <be>
Matched IS

SYNTAX ERROR: expected PERIOD but found ne
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:50:48 PM PST
```

CONNECTOR Then
ACTOR I/me
DESCRIPTION rika
TENSE IS

6- Test results-Test3

```
Script started on Tue 15 Dec 2020 04:50:54 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ tr   anslator.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest3
Processing <story>

Processing <s>
Matched CONNECTOR
Processing <noun>
Matched PRONOUN

SYNTAX ERROR: expected SUBJECT but found de
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:51:14 PM PST
```

CONNECTOR Then
ACTOR I/me
DESCRIPTION rika
TENSE IS

6- Test results - Test4

```
Script started on Tue 15 Dec 2020 04:51:28 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ tran    slator.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest4
Processing <story>

Processing <s>
Processing <noun>
Matched PRONOUN
Matched SUBJECT
Processing <afterSubject>
Processing <noun>
Matched WORD1
Processing <afterNoun>

SYNTAX ERROR: unexpected mashita found in afterNoun
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:51:47 PM PST
```

ACTOR I/me

6- Test results - Test5

```
Script started on Tue 15 Dec 2020 04:51:52 PM PST
 ]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ tran   slator.cpp
 ]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest5
Processing <story>

Processing <s>
Processing <noun>

SYNTAX ERROR: unexpected wa found in noun
 ]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:52:10 PM PST
```

6- Test results - Test6

```
Script started on Tue 15 Dec 2020 04:52:16 PM PST
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles      [?1034h[cowie001@empress
TranslatorFiles]$ g++ tran   slator.cpp
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ ./a.out
Opening File
Enter the input file name: partCtest6
Processing <story>

Lexical error: apple is not a valid token
Processing <s>
Processing <noun>

SYNTAX ERROR: unexpected apple found in noun
]0;cowie001@empress:~/cs421/cs421LK/CS421Progs/TranslatorFiles   [cowie001@empress TranslatorFiles]$ exit
exit

Script done on Tue 15 Dec 2020 04:52:34 PM PST
```