Japanese Parser Project

Group 25

By Rodolfo Rodriguez, Julian Jaramillo, and Andrew Malmstead

0- Cover sheet with the title, your group number and names.

State of the program statement

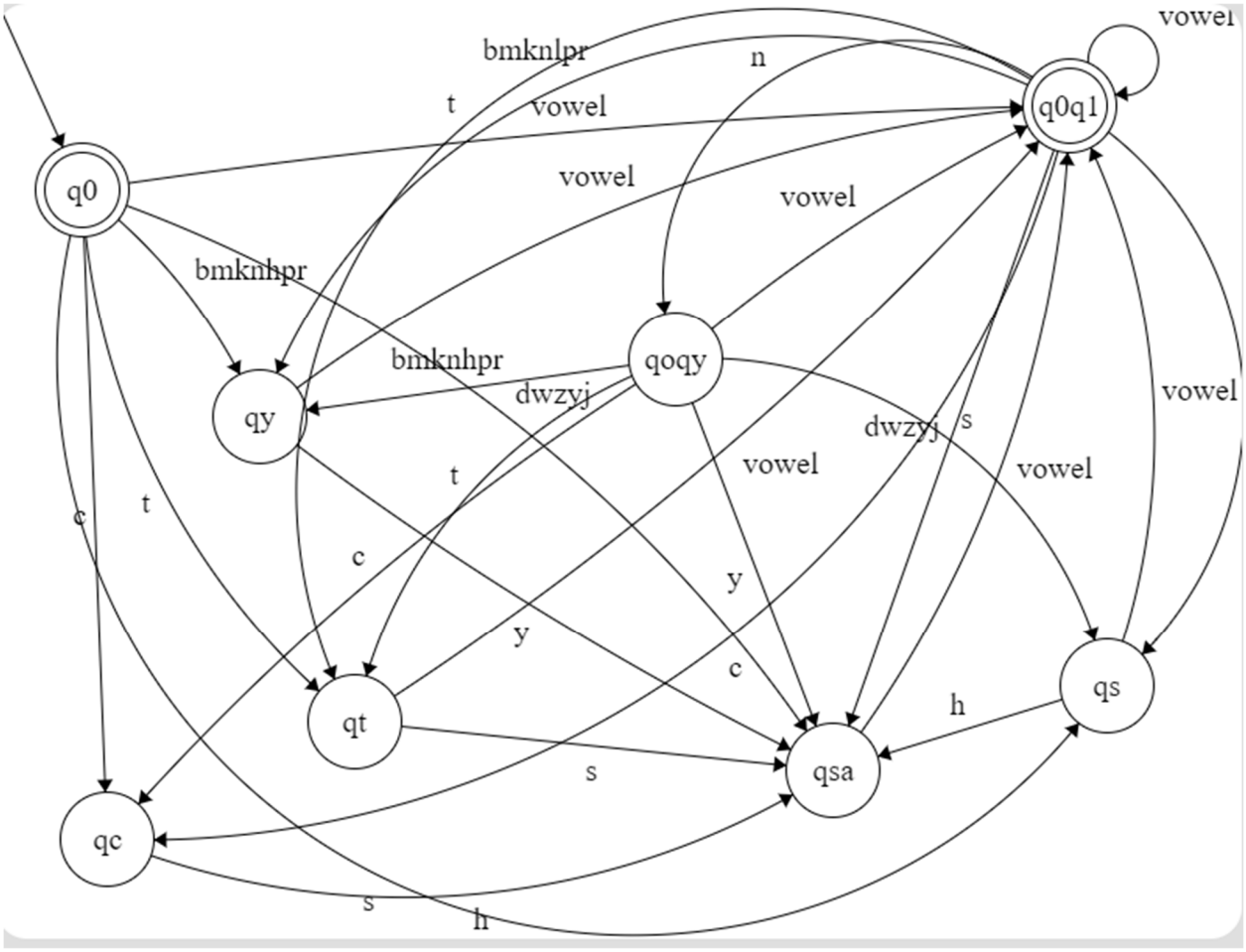
- working perfectly? yes

- any parts you did not complete? list them. All parts completed

- any bugs? list them. none

- What Extra Credit features did you implement? Give details. We implemented the tracer and syntax recovery

1- DFA (the final version)



2- Scanner code that match your DFAs (scanner.cpp)

Make sure each function has been commented as required. (regular expression; programmer names)

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

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

//=================================================

// File scanner.cpp written by: Group Number: 25

//=================================================

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

// WORD DFA
// Done by: Julian Jaramillo
// RE: (Vowel | vowel n | consonant vowel | consonant vowel n| consonant-pair vowel | consonant-pair vowel n)^+
bool word(string s)
{  int state = 0;
   int charpos = 0;
   // replace the following todo the word dfa  **
   /* 0=q0,
   1=qc
   2=qt
   3=qy
   4=qsa
   5=qs
   6=q0qy
   7=q0q1
   //q1 was dropped since nothing point it.


   */
   while (s[charpos] != '\0')
   { //q0
      if (state == 0 && s[charpos] == 'c')
         state = 1;
      else if (state == 0 && s[charpos] == 't')
         state = 2;
      else if (state == 0 && s[charpos] == 's')
         state = 5;
```

```
      else if (state == 0 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' || s[charpos] == 'k'
||
         s[charpos] == 'n' || s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
         state = 3;


      else if (state == 0 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
         s[charpos] == 'y' || s[charpos] == 'j'))
         state = 4;

      else if (state == 0 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
         state = 7;

      //qc
      else if (state == 1 && s[charpos] == 'h')
         state = 4;


      //qt
      else if (state == 2 && s[charpos] == 's')
         state = 4;
      else if (state == 2 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
         state = 7;

      //qy
      else if (state == 3 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
         state = 7;
      else if (state == 3 && s[charpos] == 'y')
         state = 4;

      //qsa
      else if (state == 4 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
         state = 7;

      //qs
      else if (state == 5 && s[charpos] == 'h')
         state = 4;
      else if (state == 5 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
```

```
            state = 7;


        //q0qy
        else if (state == 6 && s[charpos] == 'c')
            state = 1;
        else if (state == 6 && s[charpos] == 't')
            state = 2;
        else if (state == 6 && s[charpos] == 's')
            state = 5;


        else if (state == 6 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' || s[charpos] == 'k'
||
            s[charpos] == 'n' || s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
            state = 3;

        else if (state == 6 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
            s[charpos] == 'y' || s[charpos] == 'j'))
            state = 4;

        else if (state == 6 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
            s[charpos] == 'o'))
            state = 7;

        //q0q1
        else if (state == 7 && s[charpos] == 'c')
            state = 1;
        else if (state == 7 && s[charpos] == 't')
            state = 2;
        else if (state == 7 && s[charpos] == 's')
            state = 5;

        else if (state == 7 && (s[charpos] == 'n'))
            state = 6;

        else if (state == 7 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' || s[charpos] == 'k'
|| s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
            state = 3;


        else if (state == 7 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
            s[charpos] == 'y' || s[charpos] == 'j'))
            state = 4;
```

```
      else if (state == 7 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' || s[charpos] == 'E' ||
s[charpos] == 'e' || s[charpos] == 'u' ||
         s[charpos] == 'o'))
         state = 7;
      else
         return (false);
      charpos++;
   } //end of while

   // where did I end up????
   if (state == 7 || state == 6)
      return (true); // end in a final state
   else
      return (false);
}

// PERIOD DFA
// Done by: *Andrew Malmstead*
bool period(string s)
{
  if (s.compare(".") == 0)
    return true;
  else
    return false;
}

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

// TABLES Done by: **

// ** Update the tokentype to be WORD1, WORD2, PERIOD, ERROR, EOFM, etc.
enum tokentype
{
  VERB,
  VERBNEG,
  VERBPAST,
  VERBPASTNEG,
  IS,
  WAS,
  OBJECT,
  SUBJECT,
  DESTINATION,
  PRONOUN,
  CONNECTOR,
  WORD1,
  WORD2,
  EOFM,
  PERIOD,
```

```
  ERROR,
  BE,
  TENSE
};

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

string reservedWords[30] = {"masu", "masen", "mashita", "masendeshita", "desu", "deshita", "o", "wa",
"ni", "watashi", "anata", "kare", "kanojo", "sore", "mata", "soshite", "shikashi", "dakara"};

// ** 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.

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

ifstream fin; // global stream for reading from the input file

// Scanner processes only one word each time it is called
// Gives back the token type and the word itself
// ** Done by: Rodolfo Rodriguez
int scanner(tokentype &tt, string &w)
{

  // ** Grab the next word from the file via fin
  // 1. If it is eofm, return right now.
  string current;
  fin >> current;
  tt = ERROR; // setting this as a starting value for logic reasons ahead, ignore for now.

  if (current.compare(" ") == 0)
  {
    fin >> current;
  }
  if (current.compare("eofm") == 0)
  {
    //need to add token type after table is setup. tt = eofm
    w = current; //passing value of "eofm" back and breaking out of loop.
    return 0;
  }
  /*  **


  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.

***/
for (int i = 0; i < 30; i++)
{

  if (reservedWords[i] == current)
  {

    if (i >= 9)
    {
      if (i >= 13)
      {
        tt = CONNECTOR;
        w = current;
        return 0;
      }
      tt = PRONOUN;
      w = current;
      return 0;
    }
    tt = tokentype(i); //should assign correct token if both tables are setup correctly.
    w = current;
    return 0; // may delete this.
  }
}
  if (period(current))
  {
    tt = PERIOD;
    w = current;
    return 0;
  }
  else if (!(word(current)))
  {


    //condition of a double false
    tt = ERROR;  // This is to be updated after token table is created.
    w = current; // passing by reference.
    cout << "Lexical error: " << current << " is not a valid token" << endl;
    return 0; // May need to print "lexical error: &current is not a valid token" and Idea would be to
recursively call scanner(), I don't see why not.
  }

  /***

  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.


***/

 if (current.back() == 'I' || current.back() == 'E')
 {
  tt = WORD2; // needs to be added to list.
  w = current;
  return 0;
 }
 else
 {
  tt = WORD1;
  w = current;
  return 0;
 }
 cout << "this is just here for testing. End of Scanner() reached, and no token type was selected.";

 return 1; //we shouldn't get to this point and if we do its an error;
 /***
 4. Return the token type & string  (pass by reference)
 */
 // the returns are setup at all the if/else markers.
} //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)
 {
  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- Original Scanner test results

Results should be a screen dump/recordhw_LK script

- **Test 1 - with no lexical errors**
- **Test 2 - with all kinds of lexical errors**

```
Script started on Thu 10 Dec 2020 03:55:32 PM PST
malms002@empress:~/CS421Progs/ScannerFiles [?1034h[malms002@empress ScannerFiles]$ g++ -std=c++11 -o scanner scanner.cpp
malms002@empress:~/CS421Progs/ScannerFiles[malms002@empress ScannerFiles]$ ./scan
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:yarl
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;malms002@empress:~/CS421Progs/ScannerFiles[malms002@empress ScannerFiles]$ scannertest2
bash: scannertest2: command not found
 ]0;malms002@empress:~/CS421Progs/ScannerFiles[malms002@empress ScannerFiles]$ scannertes
[6P./scan
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;malms002@empress:~/CS421Progs/ScannerFiles[malms002@empress ScannerFiles]$ exit
exit

Script done on Thu 10 Dec 2020 03:57:21 PM PST

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

1. <story> ::= <s> { <s> } // stay in the loop as long as a possible start
i. // of <s> is the next_token (note it can be CONNECTOR or WORD1 or PRONOUN)
2. <s>::=[CONNECTOR] <noun> SUBJECT <after_subject>
3. <after_subject> ::= <verb> <tense> PERIOD | <noun> <after_noun>
4. <after_noun>::=<be> PERIOD | DESTINATION <verb> <tense> PERIOD | OBJECT <AFTER_OBJECT>
5. <AFTER_OBJECT>::= <verb> <tense> PERIOD |<noun> DESTINATION <verb> <tense> PERIOD
6. <noun> ::= WORD1 | PRONOUN
7. <verb> ::= WORD2
8. <be> ::= IS | WAS
9. <tense> := VERBPAST | VERBPASTNEG | VERB | VERBNEG

5- Updated Parser code for Translation (translator.cpp)

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <map>
using namespace std;
stringstream split;
/* Look for all **'s and complete them */
static std::map<std::string, std::string> dict;

bool replace;
string saved_E_word;

//=================================================
// File scanner.cpp written by: Group Number: ** //
=================================================

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

// WORD DFA
// Done by: Julian Jaramillo
// RE:(Vowel | vowel n | consonant vowel | consonant vowel n| consonant-pair vowel | consonant-pair vowel n)^+
bool word(string s)
{
	int state = 0;
	int charpos = 0;
	// replace the following todo the word dfa  **
	/* 0=q0,
	1=qc
	2=qt
	3=qy
	4=qsa
	5=qs
	6=q0qy
	7=q0q1
	//q1 was dropped since nothing point it.

	*/
	while (s[charpos] != '\0')
	{ //q0
		if (state == 0 && s[charpos] == 'c')
			state = 1;
		else if (state == 0 && s[charpos] == 't')
			state = 2;
		else if (state == 0 && s[charpos] == 's')
			state = 5;
```

```
                else if (state == 0 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' ||
s[charpos] == 'k' ||
                        s[charpos] == 'n' || s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
                        state = 3;

                else if (state == 0 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
                        s[charpos] == 'y' || s[charpos] == 'j'))
                        state = 4;

                else if (state == 0 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                        s[charpos] == 'o'))
                        state = 7;

                //qc
                else if (state == 1 && s[charpos] == 'h')
                        state = 4;

                //qt
                else if (state == 2 && s[charpos] == 's')
                        state = 4;
                else if (state == 2 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                        s[charpos] == 'o'))
                        state = 7;

                //qy
                else if (state == 3 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                        s[charpos] == 'o'))
                        state = 7;
                else if (state == 3 && s[charpos] == 'y')
                        state = 4;

                //qsa
                else if (state == 4 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                        s[charpos] == 'o'))
                        state = 7;

                //qs
                else if (state == 5 && s[charpos] == 'h')
                        state = 4;
                else if (state == 5 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                        s[charpos] == 'o'))
                        state = 7;
```

```
                    //q0qy
                    else if (state == 6 && s[charpos] == 'c')
                              state = 1;
                    else if (state == 6 && s[charpos] == 't')
                              state = 2;
                    else if (state == 6 && s[charpos] == 's')
                              state = 5;

                    else if (state == 6 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' ||
s[charpos] == 'k' ||
                              s[charpos] == 'n' || s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
                              state = 3;

                    else if (state == 6 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
                              s[charpos] == 'y' || s[charpos] == 'j'))
                              state = 4;

                    else if (state == 6 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                              s[charpos] == 'o'))
                              state = 7;

                    //q0q1
                    else if (state == 7 && s[charpos] == 'c')
                              state = 1;
                    else if (state == 7 && s[charpos] == 't')
                              state = 2;
                    else if (state == 7 && s[charpos] == 's')
                              state = 5;

                    else if (state == 7 && (s[charpos] == 'n'))
                              state = 6;

                    else if (state == 7 && (s[charpos] == 'g' || s[charpos] == 'b' || s[charpos] == 'm' ||
s[charpos] == 'k' || s[charpos] == 'h' || s[charpos] == 'p' || s[charpos] == 'r'))
                              state = 3;

                    else if (state == 7 && (s[charpos] == 'd' || s[charpos] == 'w' || s[charpos] == 'z' ||
                              s[charpos] == 'y' || s[charpos] == 'j'))
                              state = 4;

                    else if (state == 7 && (s[charpos] == 'a' || s[charpos] == 'i' || s[charpos] == 'I' ||
s[charpos] == 'E' || s[charpos] == 'e' || s[charpos] == 'u' ||
                              s[charpos] == 'o'))
                              state = 7;
                    else
                              return (false);
```

```
		charpos++;
	} //end of while

	// where did I end up????
	if (state == 7 || state == 6)
		return (true); // end in a final state
	else
		return (false);
}

// PERIOD DFA
// Done by: *Andrew Malmstead*
bool period(string s)
{
	if (s.compare(".") == 0)
		return true;
	else
		return false;
}

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

// TABLES Done by: Group

// ** Update the tokentype to be WORD1, WORD2, PERIOD, ERROR, EOFM, etc.
enum tokentype
{
	VERB,
	VERBNEG,
	VERBPAST,
	VERBPASTNEG,
	IS,
	WAS,
	OBJECT,
	SUBJECT,
	DESTINATION,
	PRONOUN,
	CONNECTOR,
	WORD1,
	WORD2,
	EOFM,
	PERIOD,
	ERROR,
	BE,
	TENSE,
	NOUN,

	STORY
```

```
};

// ** For the display names of tokens - must be in the same order as the tokentype.
string tokenName[34] = { "VERB", "VERBNEG", "VERBPAST", "VERBPASTNEG", "IS", "WAS", "OBJECT",
"SUBJECT", "DESTINATION", "PRONOUN", "CONNECTOR", "WORD1", "WORD2", "EOFM", "PERIOD",
"ERROR", "BE","TENSE", "NOUN","STORY" };

string reservedWords[18] = { "masu", "masen", "mashita", "masendeshita", "desu", "deshita", "o", "wa",
"ni", "watashi", "anata", "kare", "kanojo", "sore", "mata", "soshite", "shikashi", "dakara" };

// ** 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.

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

ifstream fin; // global stream for reading from the input file

// Scanner processes only one word each time it is called
// Gives back the token type and the word itself
// ** Done by: Rodolfo Rodriguez
int scanner(tokentype& tt, string& w)
{
	// ** Grab the next word from the file via fin
	// 1. If it is eofm, return right now.
	string current;
	if (!replace) {// checks if we're adding a word in manually through syntax error 1
		split >> current;
	}
	else {
		cout << "input a word: ";
		cin >> current;
		cout << endl;
	}
	if (current.compare(" ") == 0)
	{
		fin >> current;
	}
	if (current.compare("") == 0)
		return 1;
	tt = ERROR; // setting this as a starting value for logic reasons ahead, ignore for now.
	cout << "Scanner called using word: " << current << endl;

	if (current.compare("eofm") == 0)
	{
		//need to add token type after table is setup. tt = eofm
		w = current; //passing value of "eofm" back and breaking out of loop.
		return 0;
```

```
	}
	/*  **

	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.

 ***/

	for (int i = 0; i < 18; i++)
	{
		if (reservedWords[i] == current)
		{
			if (i >= 9)
			{
				if (i >= 13)
				{
					tt = CONNECTOR;
					w = current;
					return 0;
				}
				tt = PRONOUN;
				w = current;
				return 0;
			}
			tt = tokentype(i); //should assign correct token if both tables are setup correctly.
			w = current;
			return 0; // may delete this.
		}
	}
	if (period(current))
	{
		tt = PERIOD;
		w = current;
		return 0;
	}
	else if (!(word(current)))
	{
		//condition of a double false
		tt = ERROR;  // This is to be updated after token table is created.
		w = current; // passing by reference.
		cout << "Lexical error: " << current << " is not a valid token" << endl;
		return 0; // May need to print "lexical error: &current is not a valid token" and Idea
would be to recursively call scanner(), I don't see why not.
	}

	/***
```

```
	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.

	***/

	if (current.back() == 'I' || current.back() == 'E')
	{
		tt = WORD2; // needs to be added to list.
		w = current;
		return 0;
	}
	else
	{
		tt = WORD1;
		w = current;
		return 0;
	}
	cout << "this is just here for testing. End of Scanner() reached, and no token type was selected.";

	return 1; //we shouldn't get to this point and if we do its an error;
	/***
	4. Return the token type & string  (pass by reference)
	*/
	// the returns are setup at all the if/else markers.
} //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

void AFTER_NOUN();
/* 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 parser.cpp written by Group Number: *25*
//=================================================
```

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

// ** Need syntax_error1 and syntax_error2 functions (each takes 2 args)
//    to display syntax error messages as specified by me.

// Type of error: Error 1
// Done by: Julian
ofstream RecordErrors("errors.txt");

bool syntax_error1(string lexeme, tokentype token)
{
	cout << "SYNTAX ERROR : expected " << tokenName[token] << " but found " << lexeme << endl;
	RecordErrors << "SYNTAX ERROR: expected " << tokenName[token] << " but found " << lexeme
<< endl;

	return false;
}
// Type of error: Error 2
// Done by: Julian
void syntax_error2(tokentype input, tokentype expected)
{
	cout << "SYNTAX ERROR : expected " << tokenName[expected] << " but found " <<
tokenName[input] << endl;
	exit(1);
}

string saved_lexeme;		// the example has this within next_token()
bool token_available;		//not sure if this needs to be here.
bool display_tracing_flag = true; // used for turning on and off tracing messages
ofstream translated_file("translated.txt");

tokentype saved_token;
string checkDict(string input) {
	try {// add english work.
		dict.at(input);
		return dict.at(input);
	}
	catch (std::out_of_range) { // add japanese word
		return input;
	}
}

void createDict() {
	string filename = "lexicon.txt";
	ifstream fin2;
	string line;
	string line2;
```

```
	fin2.open(filename.c_str());
	if (!fin2.good()) {
		std::cout << "Enter the dictionary file name: ";
		cin >> filename;
		fin2.open(filename.c_str());
	}

	cout << endl;
	if (fin2.good()) {//if the file is good run the parser.
		while (fin2 >> line) {
			if (fin2 >> line2) {
				dict[line] = line2;
			}
			else
			{
				std::cout << "unbalanced dictionary file" << endl;
				break;
			}
		}
		cout << "dictionary size is: " << dict.size() << endl;
		fin2.close();
	}
}

// Purpose: takes tokentype checks for token and next
// Done by: Rudy

void getEword() {
	saved_E_word = checkDict(saved_lexeme);
}
void gen(string word)
{
	if (word != "TENSE") {
		cout << word << ": " << saved_E_word << endl;
		translated_file << word << ": " << saved_E_word << endl;
	}
	else {
		cout << word << ": " << tokenName[saved_token] << endl;
		translated_file << word << ": " << tokenName[saved_token] << endl;
	}
}

// Purpose: takes tokentype checks for token and next
// Done by: Rudy
tokentype next_token()
{
	if (!token_available)
	{
```

```
		scanner(saved_token, saved_lexeme);
		token_available = true;//We have some cache in saved_token and saved_lexeme
	}
	return saved_token;
}

// Purpose: recieves tokentype and checks if its expected type
// Done by: Rudy
bool match(tokentype expected)
{
	if (next_token() != expected)
	{
		syntax_error1(saved_lexeme, expected);

		string choice;//the user choice
		cout << "Would you like to replace or skip word? R/S ";
		cin >> choice;

		if (choice == "S" || choice == "s") {//if we choose to skip this word
			token_available = false;//make the token_available so that we can go to the
next
			return true;
		}
		else
			if (choice == "R" || choice == "r") {
				replace = true;
				scanner(saved_token, saved_lexeme);
				match(expected);
				replace = false;
			}
			else//if the user doesn't want to fix it
				exit(1);
	}
	else
	{
		token_available = false;
		//can add flag to turn on and off tracing messages
		if (display_tracing_flag == true)
			cout << " Match succeeded, token type is: " + tokenName[expected] << endl;
//display matched token_type when succeeds, used for tracing the program
	}
	return true;
}

// ----- RDP functions - one per non-term ----------
// Done by: Rudy
// Grammer:  <tense> := VERBPAST  | VERBPASTNEG | VERB | VERBNEG
void TENSE_FUNC()
```

```
{
	if (display_tracing_flag == true)
		cout << "Processing <TENSE>\n";
	switch (next_token())
	{
	case VERBPAST:
		match(VERBPAST);
		break;
	case VERBPASTNEG:
		match(VERBPASTNEG);
		break;
	case VERB:
		match(VERB);
		break;
	case VERBNEG:
		match(VERBNEG);
		break;
	default:
		syntax_error2(saved_token, TENSE);
	}
}

//Done by: Andrew
//Grammer: <verb> ::= WORD2
void VERB_FUNC()
{
	if (display_tracing_flag == true)
		cout << "Processing <VERB>\n";
	match(WORD2);
}

//Done by: Julian
// Grammer: <noun> ::= WORD1 | PRONOUN
void NOUN_FUNC()
{
	if (display_tracing_flag == true)
		cout << "Processing <NOUN>\n";
	switch (next_token())
	{
	case WORD1:
		match(WORD1);
		break;
	case PRONOUN:
		match(PRONOUN);
		break;
	default:
		syntax_error2(saved_token, NOUN);
	}
```

```
}

//Done by: Rudy
//Grammer:<after_subject> ::= <verb> <tense> PERIOD | <noun> <after_noun>
void AFTER_SUBJECT()
{
	if (display_tracing_flag == true)
		cout << "Processing <AFTER_SUBJECT>\n";
	switch (next_token())
	{
	case WORD2:
		VERB_FUNC();
		getEword();
		gen("ACTION");
		TENSE_FUNC();
		gen("TENSE");
		match(PERIOD);
		break;
	case WORD1: case PRONOUN:
		NOUN_FUNC();
		getEword();
		AFTER_NOUN();
		break;

	default:
		syntax_error2(saved_token, SUBJECT);
	}
}

//Done by: Andrew
//Grammer: <be> ::=   IS | WAS
void BE_FUNC()
{
	if (display_tracing_flag == true)
		cout << "Processing <BE>\n";
	switch (next_token())
	{
	case IS:
		match(IS);
		break;
	case WAS:
		match(WAS);
		break;
	default:
		syntax_error2(saved_token, BE);
	}
}
```

```
//Done by: Julian
//Grammer:<AFTER_OBJECT>::= <verb> <tense> PERIOD |<noun> DESTINATION <verb> <tense> PERIOD
void AFTER_OBJECT()
{
	if (display_tracing_flag == true)
		cout << "Processing <AFTER_OBJECT>\n";
	switch (next_token())
	{
	case WORD2:
		VERB_FUNC();
		getEword();
		gen("ACTION");
		TENSE_FUNC();
		gen("TENSE");
		match(PERIOD);
		break;
	case WORD1:  case PRONOUN:
		NOUN_FUNC();
		getEword();
		match(DESTINATION);
		gen("TO");
		VERB_FUNC();
		getEword();
		gen("ACTION");
		TENSE_FUNC();
		gen("TENSE");
		match(PERIOD);
		break;
	default:
		syntax_error2(saved_token, OBJECT);
	}
}

//Done by: Rudy
//Grammer:<after_noun>::=<be> PERIOD | DESTINATION <verb> <tense> PERIOD | OBJECT
void AFTER_NOUN()
{
	if (display_tracing_flag == true)
		cout << "Processing <AFTER_NOUN>\n";
	switch (next_token())
	{
	case IS: case WAS:
		gen("DESCRIPTION");
		BE_FUNC();
		gen("TENSE");
		match(PERIOD);
		break;
	case DESTINATION:
```

```
                match(DESTINATION);
                gen("TO");
                VERB_FUNC();
                getEword();
                gen("ACTION");
                TENSE_FUNC();
                gen("TENSE");
                match(PERIOD);
                break;
        case OBJECT:
                match(OBJECT);
                gen("OBJECT");
                AFTER_OBJECT();
                break;
        default:
                syntax_error2(saved_token, NOUN);
        }
}

//Done by: Andrew
//Grammer:  <s> ::= [CONNECTOR] <noun> SUBJECT <after_subject>
void s()
{
        if (display_tracing_flag == true)
                cout << "Processing <story>\n";

        switch (next_token()) {
        case CONNECTOR:
                match(CONNECTOR);
                getEword();
                gen("CONNECTOR");
                NOUN_FUNC();
                getEword();
                match(SUBJECT);
                gen("ACTOR");
                AFTER_SUBJECT();
                break;
        case PRONOUN: case WORD1:
                NOUN_FUNC();
                getEword();
                match(SUBJECT);
                gen("ACTOR");
                AFTER_SUBJECT();
                break;
        default:
                syntax_error2(saved_token, STORY);
                break;
        }
```

```
}
//Done by: Andrew
//Grammar story::<s>{<s>}
void story() {
	s();

	while (true) {
		switch (next_token())
		{
		case CONNECTOR: case PRONOUN: case WORD1:
			s();
			break;
		default:
			return;
		}
	}
}
//Done by: Julian
//Purpose: Remove the empty lines from txt file
void DeleteEmptyLines(const string& FilePath)
{
	string BufferString = "";

	//File
	fstream FileStream;
	string CurrentReadLine;

	FileStream.open(FilePath, fstream::in); //open the file in Input mode

	//Read all the lines till the end of the file
	while (getline(FileStream, CurrentReadLine))
	{
		//Check if the line is empty
		if (!CurrentReadLine.empty())
			BufferString = BufferString + CurrentReadLine + "\n";
	}
	FileStream.close();
	FileStream.open(FilePath, fstream::out); //open file in Output mode. This line will delete all data
inside the file.
	FileStream << BufferString;
	FileStream.close();
}

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

// The new test driver to start the parser
// Done by: Andrew
int main()
```

```
{
	createDict();
	std::cout << "CS 433 Programming assignment 3" << std::endl;
	std::cout << "Authors: Andrew, Rudy, and Julian" << std::endl;
	std::cout << "Date: 11/22/2020" << std::endl;
	std::cout << "Course: CS421 (Theory of Computing)" << std::endl;
	std::cout << "Description : parser project " << std::endl;
	std::cout << "================================" << std::endl;
	string choice;
	string filename;
	cout << "Display tracing messages? Y/N: ";
	cin >> choice;

	if (choice == "N" || choice == "n")
		display_tracing_flag = false;
	cout << "Enter the input file name: ";
	cin >> filename;
	ifstream fin;
	string line;
	DeleteEmptyLines(filename.c_str());
	fin.open(filename.c_str());
	int refernces = 0;
	cout << endl;
	if (fin.good()) {//if the file is good run the parser.
		while (getline(fin, line)) {//while we can get a line from the text
			refernces++;
			if (line.size() == 1)
				continue;
			if (line == "eofm")
				break;
			cout <<
"================================================================================
================" << endl;
			cout << "The line is: " << line << endl;
			split.clear();
			split << line;
			story();	//** calls the <story> to start parsing
			token_available = false;

			cout << endl;
		}
		fin.close();
	}

	if (refernces == 0)
		cout << "There is no filename:" << filename << endl;

	//close ofstream
```

```
        RecordErrors <<
"=============================================================================================
=====================================" << endl;
        RecordErrors << endl;
        translated_file <<
"=============================================================================================
=====================================" << endl;
        translated_file << endl;
        RecordErrors.close();
        translated_file.close();
}// end
```

6- Final test results:

Results should be a screen dump/recordhw_LK script with traces on.

Also, include translated.txt immediately following each script.

- Test 1 - with no errors
- Test 2-6 - with syntax errors

Extra Credit testing:

- Test 1 If you allowed trace off, Test 1 again with trace off
- Test 7 If you did syntax_error recovery, include errors.txt

```
Script started on Fri 11 Dec 2020 04:51:41 PM PST
]0;malms002@empress:~/CS421Progs/TranslatorFiles [?1034h[malms002@empress
TranslatorFiles]$ more Makefile
[?1034h[malms002@empress TranslatorFiles]$ more Makefile
####################################
# CS421 Computational Theory
# Program: A Japanese to English translator
#Features: Includes skip, and replace word, and records all errors and tranlations.
# Made by: Rodolfo Rodriguez, Julian Jaramillo, and Andrew Malmstead
####################################
CC = g++                        # use g++ for compiling c++ code or gcc for c code
CFLAGS = -O -std=c++11          # compilation flags: -Change to -O or -O2 for optimized code. -std
LIB = -lm                       # linked libraries
LDFLAGS = -L.                   # link flags
PROG = group25project                   # target executable (output)
SRCS = translator.cpp     # .c or .cpp source files.
OBJ = $(SRCS:.cpp=.o)    # object files for the target.

all : $(PROG)

$(PROG): $(OBJ)
        $(CC) -o $(PROG) $(OBJ) $(LDFLAGS) $(LIB)

.cpp.o:
        $(CC) -c $(CFLAGS) $< -o $@

depend: .depend

.depend: $(SRCS)
        rm -f ./.depend
        $(CC) $(CFLAGS) -MM $^ > ./.depend;

include .depend

# cleanup
clean:
```

```
	/bin/rm -f *.o $(PROG)

# DO NOT DELETE
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ ls
[malms002@empress TranslatorFiles]$ ls
dictionary.txt  lexicon.txt  Makefile  partCtest1  partCtest2  partCtest3  partCtest4  partCtest5  partCtest6
partctest7  partCtest7  partECtest7  @readMe  translator.cpp  ,translator.txt
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ make
[malms002@empress TranslatorFiles]$ make
g++                     -c -O -std=c++11                translator.cpp -o translator.o
g++                     -o group25project                       translator.o       -L.
-lm
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$  ls
[malms002@empress TranslatorFiles]$  ls
ls
dictionary.txt  lexicon.txt  partCtest1  partCtest3  partCtest5  partctest7  partECtest7  translator.cpp
,translator.txt
 [0m [01;32mgroup25project [0m  Makefile    partCtest2  partCtest4  partCtest6  partCtest7
@readMe     translator.o
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
 ./group25project
[malms002@empress TranslatorFiles]$  ./group25project
./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: Y
Enter the input file name: partCtest1

================================================================================
==============
The line is: watashi wa rika desu .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
```

```
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: rika
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
===============
The line is: watashi wa sensei desu .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: sensei
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: teacher
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
===============
The line is: rika wa gohan o tabE masu .
Processing <story>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: rika
Processing <AFTER_SUBJECT>
Scanner called using word: gohan
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: o
```

```
 Match succeeded, token type is: OBJECT
OBJECT: meal
Processing <AFTER_OBJECT>
Scanner called using word: tabE
Processing <VERB>
 Match succeeded, token type is: WORD2
ACTION: eat
Processing <TENSE>
Scanner called using word: masu
 Match succeeded, token type is: VERB
TENSE: VERB
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
===============
The line is: watashi wa tesuto o seito ni agE mashita .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: tesuto
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: o
 Match succeeded, token type is: OBJECT
OBJECT: test
Processing <AFTER_OBJECT>
Scanner called using word: seito
Processing <NOUN>
 Match succeeded, token type is: WORD1
Scanner called using word: ni
 Match succeeded, token type is: DESTINATION
TO: student
Processing <VERB>
Scanner called using word: agE
 Match succeeded, token type is: WORD2
ACTION: give
Processing <TENSE>
Scanner called using word: mashita
 Match succeeded, token type is: VERBPAST
TENSE: VERBPAST
Scanner called using word: .
```

```
 Match succeeded, token type is: PERIOD

================================================================================
==============
The line is: shikashi seito wa yorokobI masendeshita .
Processing <story>
Scanner called using word: shikashi
 Match succeeded, token type is: CONNECTOR
CONNECTOR: However
Processing <NOUN>
Scanner called using word: seito
 Match succeeded, token type is: WORD1
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: student
Processing <AFTER_SUBJECT>
Scanner called using word: yorokobI
Processing <VERB>
 Match succeeded, token type is: WORD2
ACTION: enjoy
Processing <TENSE>
Scanner called using word: masendeshita
 Match succeeded, token type is: VERBPASTNEG
TENSE: VERBPASTNEG
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
==============
The line is: dakara watashi wa kanashii deshita .
Processing <story>
Scanner called using word: dakara
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Therefore
Processing <NOUN>
Scanner called using word: watashi
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: kanashii
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: deshita
DESCRIPTION: sad
Processing <BE>
```

```
 Match succeeded, token type is: WAS
TENSE: WAS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
===============
The line is: soshite rika wa toire ni ikI mashita .
Processing <story>
Scanner called using word: soshite
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Then
Processing <NOUN>
Scanner called using word: rika
 Match succeeded, token type is: WORD1
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: rika
Processing <AFTER_SUBJECT>
Scanner called using word: toire
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: ni
 Match succeeded, token type is: DESTINATION
TO: restroom
Processing <VERB>
Scanner called using word: ikI
 Match succeeded, token type is: WORD2
ACTION: go
Processing <TENSE>
Scanner called using word: mashita
 Match succeeded, token type is: VERBPAST
TENSE: VERBPAST
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
===============
The line is: rika wa nakI mashita .
Processing <story>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: rika
Processing <AFTER_SUBJECT>
```

```
Scanner called using word: nakI
Processing <VERB>
 Match succeeded, token type is: WORD2
ACTION: cry
Processing <TENSE>
Scanner called using word: mashita
 Match succeeded, token type is: VERBPAST
TENSE: VERBPAST
Scanner called using word: .
 Match succeeded, token type is: PERIOD

 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: N
Enter the input file name: partCtest1

================================================================================
===============
The line is: watashi wa rika desu .
Scanner called using word: watashi
Scanner called using word: wa
ACTOR: I/me
Scanner called using word: rika
Scanner called using word: desu
DESCRIPTION: rika
TENSE: IS
Scanner called using word: .

================================================================================
===============
The line is: watashi wa sensei desu .
Scanner called using word: watashi
Scanner called using word: wa
ACTOR: I/me
Scanner called using word: sensei
Scanner called using word: desu
DESCRIPTION: teacher
TENSE: IS
Scanner called using word: .
```

```
================================================================================
===============
The line is: rika wa gohan o tabE masu .
Scanner called using word: rika
Scanner called using word: wa
ACTOR: rika
Scanner called using word: gohan
Scanner called using word: o
OBJECT: meal
Scanner called using word: tabE
ACTION: eat
Scanner called using word: masu
TENSE: VERB
Scanner called using word: .

================================================================================
===============
The line is: watashi wa tesuto o seito ni agE mashita .
Scanner called using word: watashi
Scanner called using word: wa
ACTOR: I/me
Scanner called using word: tesuto
Scanner called using word: o
OBJECT: test
Scanner called using word: seito
Scanner called using word: ni
TO: student
Scanner called using word: agE
ACTION: give
Scanner called using word: mashita
TENSE: VERBPAST
Scanner called using word: .

================================================================================
===============
The line is: shikashi seito wa yorokobI masendeshita .
Scanner called using word: shikashi
CONNECTOR: However
Scanner called using word: seito
Scanner called using word: wa
ACTOR: student
Scanner called using word: yorokobI
ACTION: enjoy
Scanner called using word: masendeshita
TENSE: VERBPASTNEG
Scanner called using word: .
```

```
================================================================================
===============
The line is: dakara watashi wa kanashii deshita .
Scanner called using word: dakara
CONNECTOR: Therefore
Scanner called using word: watashi
Scanner called using word: wa
ACTOR: I/me
Scanner called using word: kanashii
Scanner called using word: deshita
DESCRIPTION: sad
TENSE: WAS
Scanner called using word: .

================================================================================
===============
The line is: soshite rika wa toire ni ikI mashita .
Scanner called using word: soshite
CONNECTOR: Then
Scanner called using word: rika
Scanner called using word: wa
ACTOR: rika
Scanner called using word: toire
Scanner called using word: ni
TO: restroom
Scanner called using word: ikI
ACTION: go
Scanner called using word: mashita
TENSE: VERBPAST
Scanner called using word: .

================================================================================
===============
The line is: rika wa nakI mashita .
Scanner called using word: rika
Scanner called using word: wa
ACTOR: rika
Scanner called using word: nakI
ACTION: cry
Scanner called using word: mashita
TENSE: VERBPAST
Scanner called using word: .

 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
```

CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
==========
Display tracing messages? Y/N: Y
Enter the input file name: partCtest2

==========
The line is: soshite watashi wa rika desu ne .
Processing <story>
Scanner called using word: soshite
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Then
Processing <NOUN>
Scanner called using word: watashi
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: rika
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: ne
SYNTAX ERROR : expected PERIOD but found ne
Would you like to replace or skip word? R/S s
Scanner called using word: .

==========
The line is: watashi wa sensei desu .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>

```
Scanner called using word: sensei
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: teacher
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: Y
Enter the input file name: partCtest3

================================================================================
===============
The line is: dakara watashi de rika desu .
Processing <story>
Scanner called using word: dakara
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Therefore
Processing <NOUN>
Scanner called using word: watashi
 Match succeeded, token type is: PRONOUN
Scanner called using word: de
SYNTAX ERROR : expected SUBJECT but found de
Would you like to replace or skip word? R/S s
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: rika
Processing <BE>
```

```
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: Y
Enter the input file name: partCtest4

================================================================================
===============
The line is: watashi wa rika mashita .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: mashita
SYNTAX ERROR : expected NOUN but found VERBPAST
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: y
```

```
Enter the input file name: partCtest5

================================================================================
===============
The line is: wa rika desu .
Processing <story>
Scanner called using word: wa
SYNTAX ERROR : expected STORY but found SUBJECT
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: y
Enter the input file name: partCtest6

================================================================================
===============
The line is: apple wa red desu .
Processing <story>
Scanner called using word: apple
Lexical error: apple is not a valid token
SYNTAX ERROR : expected STORY but found ERROR
 ]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$
./group25project
[malms002@empress TranslatorFiles]$ ./group25project

dictionary size is: 47
CS 433 Programming assignment 3
Authors: Andrew, Rudy, and Julian
Date: 11/22/2020
Course: CS421 (Theory of Computing)
Description : parser project
================================
Display tracing messages? Y/N: Y
Enter the input file name: partCtest7^[[D^[[D^[[D^[partECtest7

================================================================================
===============
The line is: soshite watashi wa rika desu ne .
Processing <story>
Scanner called using word: soshite
```

```
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Then
Processing <NOUN>
Scanner called using word: watashi
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: rika
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: ne
SYNTAX ERROR : expected PERIOD but found ne
Would you like to replace or skip word? R/S R
input a word: .

Scanner called using word: .
 Match succeeded, token type is: PERIOD
Scanner called using word: .

================================================================================
===============
The line is: watashi wa sensei desu .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: sensei
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: teacher
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
```

```
 Match succeeded, token type is: PERIOD

================================================================================
==============
The line is: dakara watashi de rika desu .
Processing <story>
Scanner called using word: dakara
 Match succeeded, token type is: CONNECTOR
CONNECTOR: Therefore
Processing <NOUN>
Scanner called using word: watashi
 Match succeeded, token type is: PRONOUN
Scanner called using word: de
SYNTAX ERROR : expected SUBJECT but found de
Would you like to replace or skip word? R/S r
input a word: wa

Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
Scanner called using word: desu
DESCRIPTION: rika
Processing <BE>
 Match succeeded, token type is: IS
TENSE: IS
Scanner called using word: .
 Match succeeded, token type is: PERIOD

================================================================================
==============
The line is: watashi wa rika mashita .
Processing <story>
Scanner called using word: watashi
Processing <NOUN>
 Match succeeded, token type is: PRONOUN
Scanner called using word: wa
 Match succeeded, token type is: SUBJECT
ACTOR: I/me
Processing <AFTER_SUBJECT>
Scanner called using word: rika
Processing <NOUN>
 Match succeeded, token type is: WORD1
Processing <AFTER_NOUN>
```

Scanner called using word: mashita
SYNTAX ERROR : expected NOUN but found VERBPAST
]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ s
[malms002@empress TranslatorFiles]$ s

bash: s: command not found
]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ more errors.txt
[malms002@empress TranslatorFiles]$ more errors.txt

SYNTAX ERROR: expected PERIOD but found ne
SYNTAX ERROR: expected SUBJECT but found de
]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ more translated.txt
[malms002@empress TranslatorFiles]$ more translated.txt

CONNECTOR: Then
ACTOR: I/me
DESCRIPTION: rika
TENSE: IS
ACTOR: I/me
DESCRIPTION: teacher
TENSE: IS
CONNECTOR: Therefore
ACTOR: I/me
DESCRIPTION: rika
TENSE: IS
ACTOR: I/me
]0;malms002@empress:~/CS421Progs/TranslatorFiles [malms002@empress TranslatorFiles]$ exit
[malms002@empress TranslatorFiles]$ exit
exit

Script done on Fri 11 Dec 2020 04:57:17 PM PST