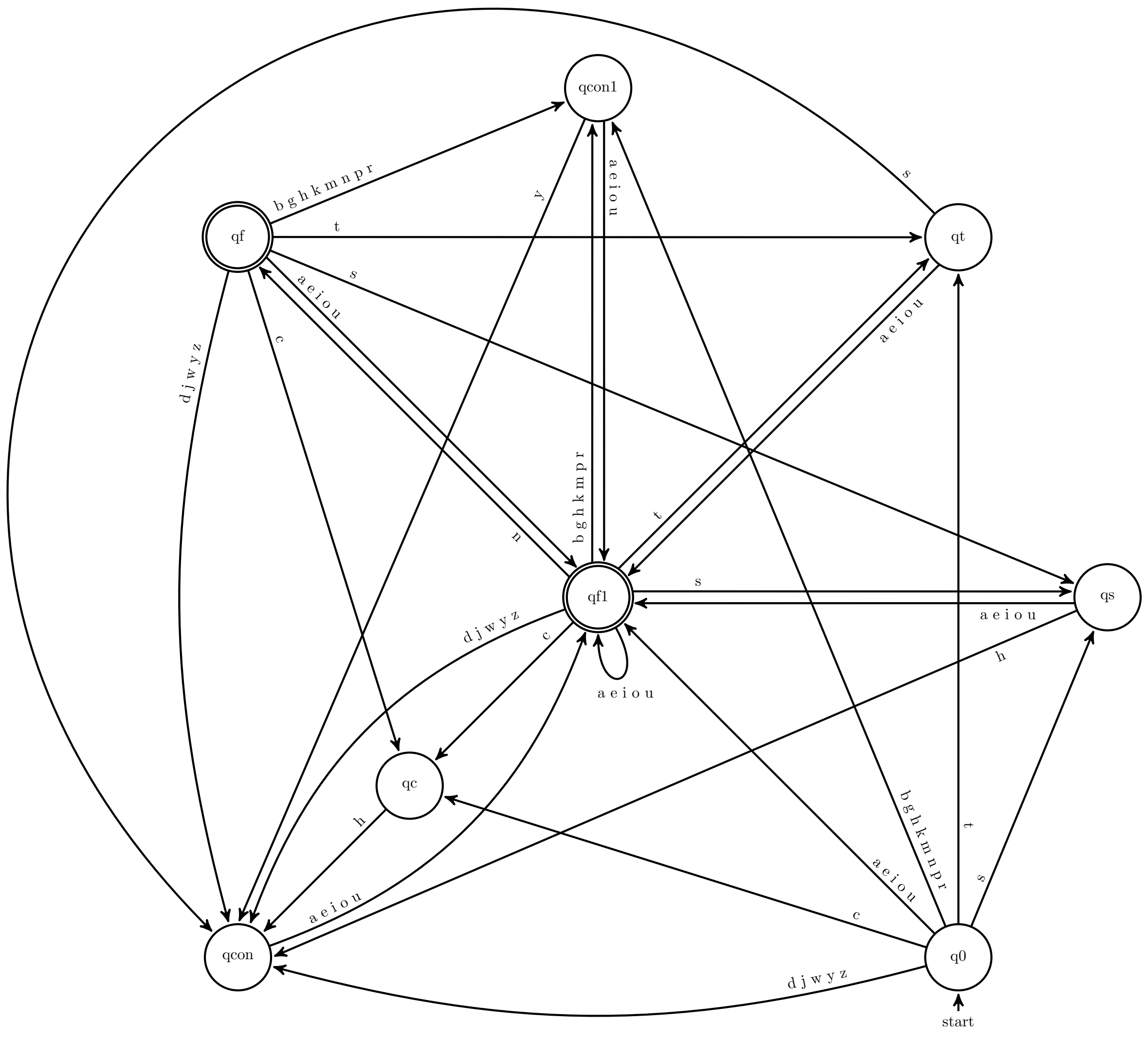
# 0 CS 421 Project

Group 12
Hugh O'Neill, Ryan Santos, Qian Zhu

State of the program:

- Working perfectly
- No incomplete parts
- No bugs
- No extra credit features

## 1 DFA


qcon1
qf
qt
qf1
qs
qc
qcon
q0
start
b g h k m n p r
t
s
a e i o u
c
d j w y z
y
n
h

## 2 Scanner Code

### 2.1 scanner.h

```
#pragma once

#include <iostream>
#include <fstream>
#include <map>
#include <string>

enum tokentype { ERROR, WORD1, WORD2, PERIOD, VERB, VERBNEG, VERBPAST, VERBPASTNEG,
                 IS, WAS, OBJECT, SUBJECT, DESTINATION, PRONOUN, CONNECTOR };

void scanner(tokentype& a, std::string& w);
```

### 2.2 scanner.cpp

```
#include "scanner.h"
using namespace std;

//=======================================================
// File scanner.cpp written by: Group Number: 12
//=======================================================

// -----  Tables -------------------------------------

string tokenName[30] = {"ERROR", "WORD1", "WORD2", "PERIOD", "VERB", "VERBNEG", "VERBPAST",
                        "VERBPASTNEG", "IS", "WAS", "OBJECT", "SUBJECT", "DESTINATION",
                        "PRONOUN", "CONNECTOR"};  // for the display names of tokens

map<string, tokentype> reserved_words = {
    {"masu", VERB},
    {"masen", VERBNEG},
    {"mashita", VERBPAST},
    {"masendeshita", VERBPASTNEG},
    {"desu", IS},
    {"deshita", WAS},
    {"o", OBJECT},
    {"wa", SUBJECT},
    {"ni", DESTINATION},
    {"watashi", PRONOUN},
    {"anata", PRONOUN},
    {"kare", PRONOUN},
    {"kanojo", PRONOUN},
    {"sore", PRONOUN},
    {"mata", CONNECTOR},
    {"soshite", CONNECTOR},
    {"shikashi", CONNECTOR},
    {"dakara", CONNECTOR},
};

// --------- DFAs ---------------------------------
// WORD DFA
// Done by: Qian Zhu
// RE: (vowel | vowel n | consonant vowel | consonant vowel n | consonant-pair vowel |
//      consonant-pair vowel n)^+
bool word(string s)
{
    /*
    q0 = 0, qc = 1, qcon = 2, qcon1 = 3,
    qs = 4, qt = 5, qf = 6, qf1 = 7
    */

    int state = 0;

    for (char c : s) {
        c = tolower(c);

        if ((state == 0 || state == 6 || state == 7) &&
            (c == 'd' || c == 'j' || c == 'w' || c == 'y' || c == 'z')) {
            state = 2;
        }
        else if ((state == 1 || state == 4) && (c == 'h')) {
            state = 2;
        }
        else if ((state == 5) && (c == 's')) {
            state = 2;
        }
        else if ((state == 3) && (c == 'y')) {
            state = 2;
        }
        else if ((state == 0 || state == 6 || state == 7) && (c == 'c')) {
            state = 1;
        }
        else if ((state == 7) && (c == 'n')) {
            state = 6;
        }
        else if ((state == 0 || state == 6) &&
            (c == 'b' || c == 'g' || c == 'h' || c == 'k' || c == 'm' || c == 'n' ||
             c == 'p' || c == 'r')) {
            state = 3;
        }
        else if ((state == 7) &&
            (c == 'b' || c == 'g' || c == 'h' || c == 'k' || c == 'm' || c == 'p' ||
             c == 'r')) {
            state = 3;
        }
        else if ((state == 0 || state == 6 || state == 7) && (c == 't')) {
            state = 5;
        }
        else if ((state == 0 || state == 6 || state == 7) && (c == 's')) {
            state = 4;
        }
        else if ((state == 0 || state == 2 || state == 3 || state == 4 || state == 5 ||
                  state == 6 || state == 7) &&
            (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')) {
```

```
            state = 7;
        }
        else {
            return false;
        }
    }

    return state == 6 || state == 7;
}

// PERIOD DFA
// Done by: Qian Zhu
bool period(string s)
{
    int state = 0;

    for (char c : s) {
        if (state == 0 && c == '.') {
            state = 1;
        }
        else {
            return false;
        }
    }

    return state == 1;
}

// ------------ Scanner -----------------------

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: Ryan Santos, Hugh O'Neill
void scanner(tokentype& a, string& w)
{
    fin >> w;  // Grab the next word from the file via fin
    cout << "Scanner called using word: " << w << endl;
    map<string, tokentype>::iterator word_type;

    if (word(w)) {
        word_type = reserved_words.find(w);
        if (word_type != reserved_words.end()) {  // reserved_words[w] exists
            a = word_type->second;
        }
        else {
            char last_c = w.back();
            a = (last_c == 'I' || last_c == 'E') ? WORD2 : WORD1;
        }
    }
    else if (period(w)) {
        a = PERIOD;
    }
    else if (w == "eofm") {
        return;
    }
    else {
        cout << endl << "Lexical error: " << w << " is not a valid token" << endl;
        a = ERROR;
    }

}//the end of scanner
```

## 3 Scanner Test Results

### 3.1 Test 1

```
[santo106@empress ScannerFiles]$ ./group12scanner.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.
```

## 3.2 Test 2

```
[santo106@empress ScannerFiles]$ ./group12scanner.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
Type is:WORD1
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.
```

## 4 Factored Rules

```
<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
```

## 5 Parser and Translator

### 5.1 translator.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <map>
#include <sstream>
#include <string>
#include "scanner.h"

// forward-declare non-terminal and translation functions
void story();
void s();
void noun();
void after_subject();
void verb();
void tense();
void after_noun();
void be();
void after_object();
void getEword();
void gen(std::string line_type);
```

### 5.2 translator.cpp

```
#include "translator.h"
using namespace std;

//=================================================
// File translator.cpp written by Group Number: 12
//=================================================

// ----- Utility and Globals -----------------------------------

extern string tokenName[];
extern map<string, tokentype> reserved_words;
extern ifstream fin;

tokentype saved_token;
bool token_available;
string saved_lexeme;
string saved_e_word;

// dictionary that will hold the content of lexicon.txt
map<string, string> lexicon;
ofstream trans;
```

```
// Done by: Ryan Santos
void syntax_error1(tokentype expected) {
    cout << endl << "SYNTAX ERROR: expected " << tokenName[expected] << " but found " <<
            saved_lexeme << endl;
    exit(1);
}

// Done by: Ryan Santos
void syntax_error2(string function_name) {
    cout << endl << "SYNTAX ERROR: unexpected " << saved_lexeme << " found in " <<
            function_name << endl;
    exit(1);
}

// Done by: Hugh O'Neill
tokentype next_token()
{
    if (!token_available) {
        scanner(saved_token, saved_lexeme);
        token_available = true;
    }

    return saved_token;
}

// Done by: Hugh O'Neill
bool match(tokentype expected)
{
    if (next_token() != expected) {
        syntax_error1(expected);
    }
    else {
        token_available = false;
        cout << "Matched " << tokenName[expected] << endl;
        return true;
    }
}

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

// Grammar: <story> ::= <s> { <s> }
// Done by: Ryan Santos
void story()
{
    cout << "Processing <story>" << endl << endl;

    s();

    bool done = false;
    while (!done) {
        switch (next_token()) {
        case CONNECTOR:
        case WORD1:
        case PRONOUN:
            s();
            break;
        default: done = true;
        }
    }

    cout << endl << "Successfully parsed <story>." << endl;
}

// Grammar: <s> ::= [CONNECTOR #getEword# #gen("CONNECTOR")#] <noun> #getEword# SUBJECT
//              #gen("ACTOR")# <after subject>
// Done by: Hugh O'Neill
void s()
{
    cout << "Processing <s>" << endl;

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

    noun();

    match(SUBJECT);
    gen("ACTOR");

    after_subject();
}

// Grammar: <noun> ::= WORD1 | PRONOUN
// Done by: Qian Zhu
void noun()
{
    cout << "Processing <noun>" << endl;

    switch (next_token()) {
    case WORD1:
        match(WORD1);
        break;
    case PRONOUN:
        match(PRONOUN);
        break;
    default:
        syntax_error2("noun");
    }

    getEword();
}

// Grammar: <after subject> ::= <verb> #getEword# #gen("ACTION")# <tense> #gen("TENSE")# PERIOD |
//              <noun> #getEword# <after noun>
// Done by: Ryan Santos
void after_subject()
```

```
{
    cout << "Processing <after_subject>" << endl;

    switch (next_token()) {
    case WORD2:
        verb();
        tense();
        match(PERIOD);
        break;
    case WORD1:
    case PRONOUN:
        noun();
        after_noun();
        break;
    default:
        syntax_error2("after_subject");
    }
}

// Grammar: <verb> ::= WORD2
// Done by: Hugh O'Neill
void verb()
{
    cout << "Processing <verb>" << endl;

    match(WORD2);
    getEword();
    gen("ACTION");
}

// Grammar: <tense> := VERBPAST | VERBPASTNEG | VERB | VERBNEG
// Done by: Qian Zhu
void tense()
{
    cout << "Processing <tense>" << endl;

    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("tense");
    }

    gen("TENSE");
}

// Grammar: <after noun> ::= <be> #gen("DESCRIPTION")# #gen("TENSE")# PERIOD |
//              DESTINATION #gen("TO")# <verb> #getEword# #gen("ACTION")# <tense> #gen("TENSE")#
                PERIOD | OBJECT #gen("OBJECT")# <after object>
// Done by: Ryan Santos
void after_noun()
{
    cout << "Processing <after_noun>" << endl;

    switch (next_token()) {
    case IS:
    case WAS:
        be();
        match(PERIOD);
        break;
    case DESTINATION:
        match(DESTINATION);
        gen("TO");
        verb();
        tense();
        match(PERIOD);
        break;
    case OBJECT:
        match(OBJECT);
        gen("OBJECT");
        after_object();
        break;
    default:
        syntax_error2("after_noun");
    }
}

// Grammar: <be> ::= IS | WAS
// Done by: Hugh O'Neill
void be()
{
    cout << "Processing <be>" << endl;

    switch (next_token()) {
    case IS:
        match(IS);
        break;
    case WAS:
        match(WAS);
        break;
    default:
        syntax_error2("be");
    }

    gen("DESCRIPTION");
    gen("TENSE");
}
```

```
// Grammar: <after object> ::= <verb> #getEword# #gen("ACTION")# <tense> #gen("TENSE")# PERIOD |
//              <noun> #getEword# DESTINATION #gen("TO")# <verb> #getEword# #gen("ACTION")#
//              <tense> #gen("TENSE")# PERIOD
// Done by: Qian Zhu
void after_object()
{
    cout << "Processing <after_object>" << endl;

    switch (next_token()) {
    case WORD2:
        verb();
        tense();
        match(PERIOD);
        break;
    case WORD1:
    case PRONOUN:
        noun();
        match(DESTINATION);
        gen("TO");
        verb();
        tense();
        match(PERIOD);
        break;
    default:
        syntax_error2("after_object");
    }
}

//---------------------------------------

// using the current lexeme, look up the English word in the Lexicon if it is there
// save the result in saved_E_word
// otherwise, save the Japanese word as-is
// Done by: Ryan Santos
void getEword()
{
    map<string, string>::iterator e_word = lexicon.find(saved_lexeme);
    if (e_word != lexicon.end()) {  // lexicon[saved_lexeme] exists
        saved_e_word = e_word->second;
    }
    else {
        saved_e_word = saved_lexeme;
    }
}

// using the line type, sends a line of an IR to translated.txt
// saved_E_word or saved_token is used
// Done by: Hugh O'Neill
void gen(string line_type)
{
    string saved;

    if (line_type == "TENSE") {
        saved = tokenName[saved_token] + '\n';
    }
    else {
        saved = saved_e_word;
    }

    trans << line_type << ": " << saved << endl;
}

// ------------------------------------------

// The final test driver to start the translator
// Done by: Qian Zhu
int main()
{
    // open the lexicon.txt file and read it in
    ifstream lex("lexicon.txt", ios::in);
    string line;

    while (getline(lex, line)) {
        istringstream tokens(line);
        string jap, eng;

        tokens >> jap;
        tokens >> eng;

        lexicon[jap] = eng;
    }

    lex.close();

    trans.open("translated.txt", ios::out);

    string filename;

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

    story();  // start parsing

    fin.close();
    trans.close();
}// end
```

## 6 Final Test Results

### 6.1 Test 1

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest1
Processing <story>

Processing <s>
Scanner called using word: watashi
```

```
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: rika
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: desu
Processing <be>
Matched IS
Scanner called using word: .
Matched PERIOD
Scanner called using word: watashi
Processing <s>
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: sensei
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: desu
Processing <be>
Matched IS
Scanner called using word: .
Matched PERIOD
Scanner called using word: rika
Processing <s>
Processing <noun>
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: gohan
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: o
Matched OBJECT
Processing <after_object>
Scanner called using word: tabE
Processing <verb>
Matched WORD2
Processing <tense>
Scanner called using word: masu
Matched VERB
Scanner called using word: .
Matched PERIOD
Scanner called using word: watashi
Processing <s>
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: tesuto
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: o
Matched OBJECT
Processing <after_object>
Scanner called using word: seito
Processing <noun>
Matched WORD1
Scanner called using word: ni
Matched DESTINATION
Processing <verb>
Scanner called using word: agE
Matched WORD2
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: shikashi
Processing <s>
Matched CONNECTOR
Processing <noun>
Scanner called using word: seito
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: yorokobI
Processing <verb>
Matched WORD2
Processing <tense>
Scanner called using word: masendeshita
Matched VERBPASTNEG
Scanner called using word: .
Matched PERIOD
Scanner called using word: dakara
Processing <s>
Matched CONNECTOR
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: kanashii
Processing <noun>
Matched WORD1
```

```
Processing <after_noun>
Scanner called using word: deshita
Processing <be>
Matched WAS
Scanner called using word: .
Matched PERIOD
Scanner called using word: soshite
Processing <s>
Matched CONNECTOR
Processing <noun>
Scanner called using word: rika
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: toire
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: ni
Matched DESTINATION
Processing <verb>
Scanner called using word: ikI
Matched WORD2
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: rika
Processing <s>
Processing <noun>
Matched WORD1
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: nakI
Processing <verb>
Matched WORD2
Processing <tense>
Scanner called using word: mashita
Matched VERBPAST
Scanner called using word: .
Matched PERIOD
Scanner called using word: eofm

Successfully parsed <story>.
[santo106@empress TranslatorFiles]$ cat translated.txt
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
```

## 6.2 Test 2

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest2
Processing <story>

Processing <s>
Scanner called using word: soshite
Matched CONNECTOR
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: rika
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: desu
Processing <be>
```

```
Matched IS
Scanner called using word: ne

SYNTAX ERROR: expected PERIOD but found ne
[santo106@empress TranslatorFiles]$ cat translated.txt
CONNECTOR: Then
ACTOR: I/me
DESCRIPTION: rika
TENSE: IS
```

## 6.3 Test 3

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest3
Processing <story>

Processing <s>
Scanner called using word: dakara
Matched CONNECTOR
Processing <noun>
Scanner called using word: watashi
Matched PRONOUN
Scanner called using word: de

SYNTAX ERROR: expected SUBJECT but found de
[santo106@empress TranslatorFiles]$ cat translated.txt
CONNECTOR: Therefore
```

## 6.4 Test 4

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest4
Processing <story>

Processing <s>
Scanner called using word: watashi
Processing <noun>
Matched PRONOUN
Scanner called using word: wa
Matched SUBJECT
Processing <after_subject>
Scanner called using word: rika
Processing <noun>
Matched WORD1
Processing <after_noun>
Scanner called using word: mashita

SYNTAX ERROR: unexpected mashita found in after_noun
[santo106@empress TranslatorFiles]$ cat translated.txt
ACTOR: I/me
```

### 6.5 Test 5

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest5
Processing <story>

Processing <s>
Scanner called using word: wa
Processing <noun>

SYNTAX ERROR: unexpected wa found in noun
[santo106@empress TranslatorFiles]$ cat translated.txt
```

### 6.6 Test 6

```
[santo106@empress TranslatorFiles]$ ./group12project.out
Enter the input file name: partCtest6
Processing <story>

Processing <s>
Scanner called using word: apple

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

SYNTAX ERROR: unexpected apple found in noun
[santo106@empress TranslatorFiles]$ cat translated.txt
```