Jason Martin

Christopher Cervantes-Hernandez

Efren Martinez

Working perfectly?

**Yes**

Any parts you did not complete?

**No**

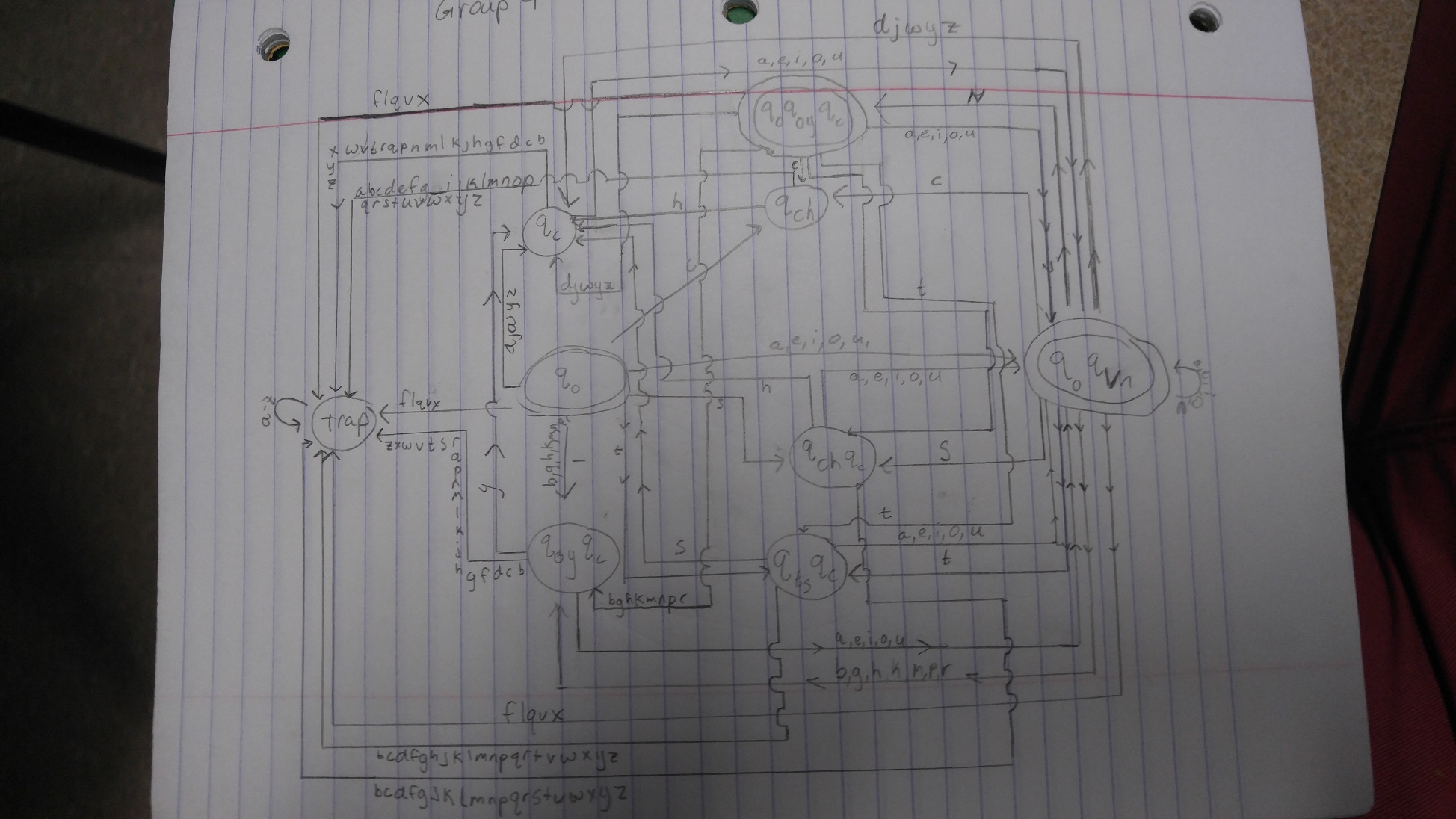
Any bugs?

**None that we've noticed**

What extra credit features did you implement?

**The option to turn trace off by asking the user after the file name has been entered. Their response('y' or 'n') is saved and checked whenever it comes across a trace message in the code.**

**Saving all past and present error messages in errors.txt, along with the name of the file where the error was found, by writing to file at the same type the message is printed on-screen.**

****

**#include<iostream>**

**#include<fstream>**

**#include<string>**

**#include<vector>**

**using namespace std;**

**//=====================================================**

**// File scanner.cpp written by: Group Number: 4**

**//=====================================================**

**// \*\* MYTOKEN DFA to be replaced by the WORD DFA**

**// \*\* Done by: Efren Martinez**

**// \*\* RE: (vowel | vowel n | consonant vowel | consonant vowel n | consonant-pair vowel | consonant-pair vowel n)^+**

**bool wordToken(string s)**

**{**

**string state = "q0";**

**for(int pos = 0; pos < s.size(); pos++)**

**{**

**if(state == "q0")**

**{**

**//Since 'e' and 'i' can be upper-case, tolower() is used when checking those letters**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 'b' || s[pos] == 'g' || s[pos] == 'h' || s[pos] == 'k' || s[pos] == 'm' || s[pos] == 'n' || s[pos] == 'p' || s[pos] == 'r')**

**state = "q0yqc";**

**else if(s[pos] == 'd' || s[pos] == 'j' || s[pos] == 'w' || s[pos] == 'y' || s[pos] == 'z')**

**state = "qc";**

**else if(s[pos] == 't')**

**state = "qtsqc";**

**else if(s[pos] == 's')**

**state = "qchqc";**

**else if(s[pos] == 'c')**

**state = "qch";**

**else //all letters not already checked go to state "trap", ie. it fails and returns false**

**return false;**

**}**

**else if(state == "q0qVn")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 'b' || s[pos] == 'g' || s[pos] == 'h' || s[pos] == 'k' || s[pos] == 'm' || s[pos] == 'p' || s[pos] == 'r')**

**state = "q0yqc";**

**else if(s[pos] == 'n')**

**state = "q0q0yqc";**

**else if(s[pos] == 'd' || s[pos] == 'j' || s[pos] == 'w' || s[pos] == 'y' || s[pos] == 'z')**

**state = "qc";**

**else if(s[pos] == 't')**

**state = "qtsqc";**

**else if(s[pos] == 's')**

**state = "qchqc";**

**else if(s[pos] == 'c')**

**state = "qch";**

**else**

**return false;**

**}**

**else if(state == "q0yqc")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 'y')**

**state = "qc";**

**else**

**return false;**

**}**

**else if(state == "qc")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else**

**return false;**

**}**

**else if(state == "qtsqc")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 's')**

**state = "qc";**

**else**

**return false;**

**}**

**else if(state == "qchqc")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 'h')**

**state = "qc";**

**else**

**return false;**

**}**

**else if(state == "qch")**

**{**

**if(s[pos] == 'h')**

**state = "qc";**

**else**

**return false;**

**}**

**else if(state == "q0q0yqc")**

**{**

**if(s[pos] == 'a' || tolower(s[pos]) == 'e' || tolower(s[pos]) == 'i' || s[pos] == 'o' || s[pos] == 'u')**

**state = "q0qVn";**

**else if(s[pos] == 'b' || s[pos] == 'g' || s[pos] == 'h' || s[pos] == 'k' || s[pos] == 'm' || s[pos] == 'n' || s[pos] == 'p' || s[pos] == 'r')**

**state = "q0yqc";**

**else if(s[pos] == 'd' || s[pos] == 'j' || s[pos] == 'w' || s[pos] == 'y' || s[pos] == 'z')**

**state = "qc";**

**else if(s[pos] == 't')**

**state = "qtsqc";**

**else if(s[pos] == 's')**

**state = "qchqc";**

**else if(s[pos] == 'c')**

**state = "qch";**

**else**

**return false;**

**}**

**}**

**//if in finals states "q0qVn" or "q0q0yqc" then return true**

**if(state == "q0qVn" || state == "q0q0yqc")**

**return true;**

**else**

**return false;**

**}**

**// \*\* Add the PERIOD DFA here**

**// \*\* Done by: Efren Martinez**

**bool punctuationToken(string s)**

**{**

**if(s == ".")**

**return true;**

**else**

**return false;**

**}**

**// \*\* Update the tokentype to be WORD1, WORD2, PERIOD and ERROR.**

**typedef enum tokentype {WORD1, WORD2, PERIOD, ERROR,**

**VERB,VERBNEG,VERBPAST,VERBPASTNEG,**

**IS,WAS,OBJECT,SUBJECT,DESTINATION,**

**PRONOUN, CONNECTOR, EOFM };**

**// \*\* Need the lexicon to be set up here (to be used in Part C)**

**// \*\* Need the reservedwords list to be set up here**

**// \*\* Done by: Jason and Chris**

**bool isReserved(string s, tokentype& t)**

**{**

**for(int i = 0; i < s.size(); i++)**

**{**

**s[i] = tolower(s[i]);**

**}**

**if(s == "masu")**

**{**

**t = VERB;**

**return true;**

**}**

**else if(s == "masen")**

**{**

**t = VERBNEG;**

**return true;**

**}**

**else if(s == "mashita")**

**{**

**t = VERBPAST;**

**return true;**

**}**

**else if(s == "masendeshita")**

**{**

**t = VERBPASTNEG;**

**return true;**

**}**

**else if(s == "desu")**

**{**

**t = IS;**

**return true;**

**}**

**else if(s == "deshita")**

**{**

**t = WAS;**

**return true;**

**}**

**else if(s == "o")**

**{**

**t = OBJECT;**

**return true;**

**}**

**else if(s == "wa")**

**{**

**t = SUBJECT;**

**return true;**

**}**

**else if(s == "ni")**

**{**

**t = DESTINATION;**

**return true;**

**}**

**else if(s == "watashi")**

**{**

**t = PRONOUN;**

**return true;**

**}**

**else if(s == "anata")**

**{**

**t = PRONOUN;**

**return true;**

**}**

**else if(s == "kare")**

**{**

**t = PRONOUN;**

**return true;**

**}**

**else if(s == "kanojo")**

**{**

**t = PRONOUN;**

**return true;**

**}**

**else if(s == "sore")**

**{**

**t = PRONOUN;**

**return true;**

**}**

**else if(s == "mata")**

**{**

**t = CONNECTOR;**

**return true;**

**}**

**else if(s == "soshite")**

**{**

**t = CONNECTOR;**

**return true;**

**}**

**else if(s == "shikashi")**

**{**

**t = CONNECTOR;**

**return true;**

**}**

**else if(s == "dakara")**

**{**

**t = CONNECTOR;**

**return true;**

**}**

**else if(s == "eofm")**

**{**

**t = EOFM;**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**// \*\* Do not require any file input for these.**

**// \*\* a.out should work without any additional files.**

**// Scanner processes only one word each time it is called**

**// \*\* Done by: Jason**

**int scanner(tokentype& a, string& w)**

**{**

**// \*\* Grab the next word from the file**

**if(punctuationToken(w))**

**a = PERIOD;**

**else if(wordToken(w))**

**{**

**if(isReserved(w,a))**

**{**

**return 1;**

**}**

**else**

**{**

**if(w[w.size()-1] == 'I' | w[w.size()-1] == 'E')**

**{**

**a = WORD2;**

**}**

**else**

**{**

**a = WORD1;**

**}**

**}**

**}**

**else**

**{**

**a = ERROR;**

**}**

**/\***

**2. Call the token functions one after another (if-then-else)**

**And generate a lexical error if both DFAs failed.**

**Let the token\_type be ERROR in that case.**

**3. Make sure WORDs are checked against the reservedwords list**

**If not reserved, token\_type is WORD1 or WORD2.**

**4. Return the token type & string (pass by reference)**

**\*/**

**return 1;**

**}//the end**

**// The test driver to call the scanner repeatedly**

**// \*\* Done by: Chris**

**int main()**

**{**

**tokentype thetype;**

**string theword;**

**string thefile;**

**fstream fin;**

**cout << "Please Enter the name of the file" << endl;**

**cin >> thefile;**

**fin.open(thefile.c\_str());**

**/\***

**1. get the input file name from the user**

**2. open the input file which contains a story written in Japanese (fin.open).**

**3. call Scanner repeatedly until the EOF marker is read, and**

**each time cout the returned results**

**e.g. STRING TOKEN-TYPE**

**===== ===========**

**watashi PRONOUN (from the first call)**

**wa SUBJECT (from the second call)**

**gakkou WORD1**

**etc.**

**\*/**

**while ( theword != "eofm")**

**{**

**fin >> theword;**

**scanner(thetype, theword); // call the scanner**

**cout << "Word is:" << theword << endl;**

**switch (thetype)**

**{**

**case WORD1:**

**cout << "Type is: WORD1" << endl;**

**break;**

**case WORD2:**

**cout << "Type is: WORD2" << endl;**

**break;**

**case PERIOD:**

**cout << "Type is: PERIOD" << endl;**

**break;**

**case ERROR:**

**cout << "Lexical error: " << theword <<" is not a valid token" << endl;**

**break;**

**case VERB:**

**cout << "Type is: VERB" << endl;**

**break;**

**case VERBNEG:**

**cout << "Type is: VERBNEG" << endl;**

**break;**

**case VERBPAST:**

**cout << "Type is: VERBPAST" << endl;**

**break;**

**case VERBPASTNEG:**

**cout << "Type is: VERBPASTNEG" << endl;**

**break;**

**case IS:**

**cout << "Type is: IS" << endl;**

**break;**

**case WAS:**

**cout << "Type is: WAS" << endl;**

**break;**

**case OBJECT:**

**cout << "Type is: OBJECT" << endl;**

**break;**

**case SUBJECT:**

**cout << "Type is: SUBJECT" << endl;**

**break;**

**case DESTINATION:**

**cout << "Type is: DESTINATION" << endl;**

**break;**

**case PRONOUN:**

**cout << "Type is: PRONOUN" << endl;**

**break;**

**case CONNECTOR:**

**cout << "Type is: CONNECTOR" << endl;**

**break;**

**case EOFM:**

**break;**

**};**

**// \*\* display the actual type instead of a number**

**}**

**// \*\* close the input file**

**}// end**

**Script started on Wed 06 Dec 2017 03:58:43 PM PST**

**]0;marti545@empress:~/cs421/CS421Progs/ScannerFiles[marti545@empress ScannerFiles]$ ./a.out**

**Please Enter the name of the file**

**partBtesscannertest1**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:rika**

**Type is: WORD1**

**Word is:desu**

**Type is: IS**

**Word is:.**

**Type is: PERIOD**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:sensei**

**Type is: WORD1**

**Word is:desu**

**Type is: IS**

**Word is:.**

**Type is: PERIOD**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:ryouri**

**Type is: WORD1**

**Word is:o**

**Type is: OBJECT**

**Word is:yarI**

**Type is: WORD2**

**Word is:masu**

**Type is: VERB**

**Word is:.**

**Type is: PERIOD**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:gohan**

**Type is: WORD1**

**Word is:o**

**Type is: OBJECT**

**Word is:seito**

**Type is: WORD1**

**Word is:ni**

**Type is: DESTINATION**

**Word is:agE**

**Type is: WORD2**

**Word is:mashita**

**Type is: VERBPAST**

**Word is:.**

**Type is: PERIOD**

**Word is:shikashi**

**Type is: CONNECTOR**

**Word is:seito**

**Type is: WORD1**

**Word is:wa**

**Type is: SUBJECT**

**Word is:yorokobI**

**Type is: WORD2**

**Word is:masendeshita**

**Type is: VERBPASTNEG**

**Word is:.**

**Type is: PERIOD**

**Word is:dakara**

**Type is: CONNECTOR**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:kanashii**

**Type is: WORD1**

**Word is:deshita**

**Type is: WAS**

**Word is:.**

**Type is: PERIOD**

**Word is:soshite**

**Type is: CONNECTOR**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:toire**

**Type is: WORD1**

**Word is:ni**

**Type is: DESTINATION**

**Word is:ikI**

**Type is: WORD2**

**Word is:mashita**

**Type is: VERBPAST**

**Word is:.**

**Type is: PERIOD**

**Word is:watashi**

**Type is: PRONOUN**

**Word is:wa**

**Type is: SUBJECT**

**Word is:nakI**

**Type is: WORD2**

**Word is:mashita**

**Type is: VERBPAST**

**Word is:.**

**Type is: PERIOD**

**Word is:eofm**

**Lexical error: eofm is not a valid token**

**]0;marti545@empress:~/cs421/CS421Progs/ScannerFiles[marti545@empress ScannerFiles]$ ./a.out**

**Please Enter the name of the file**

**scannertest2**

**Word is:daigaku**

**Type is: WORD1**

**Word is:college**

**Lexical error: college is not a valid token**

**Word is:kurasu**

**Type is: WORD1**

**Word is:class**

**Lexical error: class is not a valid token**

**Word is:hon**

**Type is: WORD1**

**Word is:book**

**Lexical error: book is not a valid token**

**Word is:tesuto**

**Type is: WORD1**

**Word is:test**

**Lexical error: test is not a valid token**

**Word is:ie**

**Type is: WORD1**

**Word is:home\***

**Lexical error: home\* is not a valid token**

**Word is:isu**

**Type is: WORD1**

**Word is:chair**

**Lexical error: chair is not a valid token**

**Word is:seito**

**Type is: WORD1**

**Word is:student**

**Lexical error: student is not a valid token**

**Word is:sensei**

**Type is: WORD1**

**Word is:teacher**

**Lexical error: teacher is not a valid token**

**Word is:tomodachi**

**Type is: WORD1**

**Word is:friend**

**Lexical error: friend is not a valid token**

**Word is:jidoosha**

**Type is: WORD1**

**Word is:car**

**Lexical error: car is not a valid token**

**Word is:gyuunyuu**

**Type is: WORD1**

**Word is:milk**

**Lexical error: milk is not a valid token**

**Word is:sukiyaki**

**Type is: WORD1**

**Word is:tenpura**

**Type is: WORD1**

**Word is:sushi**

**Type is: WORD1**

**Word is:biiru**

**Type is: WORD1**

**Word is:beer**

**Lexical error: beer is not a valid token**

**Word is:sake**

**Type is: WORD1**

**Word is:tokyo**

**Type is: WORD1**

**Word is:kyuushuu**

**Type is: WORD1**

**Word is:Osaka**

**Lexical error: Osaka is not a valid token**

**Word is:choucho**

**Type is: WORD1**

**Word is:butterfly**

**Lexical error: butterfly is not a valid token**

**Word is:an**

**Type is: WORD1**

**Word is:idea**

**Type is: WORD1**

**Word is:yasashii**

**Type is: WORD1**

**Word is:easy**

**Lexical error: easy is not a valid token**

**Word is:muzukashii**

**Type is: WORD1**

**Word is:difficult**

**Lexical error: difficult is not a valid token**

**Word is:ureshii**

**Type is: WORD1**

**Word is:pleased**

**Lexical error: pleased is not a valid token**

**Word is:shiawase**

**Type is: WORD1**

**Word is:happy**

**Lexical error: happy is not a valid token**

**Word is:kanashii**

**Type is: WORD1**

**Word is:sad**

**Lexical error: sad is not a valid token**

**Word is:omoi**

**Type is: WORD1**

**Word is:heavy**

**Lexical error: heavy is not a valid token**

**Word is:oishii**

**Type is: WORD1**

**Word is:delicious**

**Lexical error: delicious is not a valid token**

**Word is:tennen**

**Type is: WORD1**

**Word is:natural**

**Lexical error: natural is not a valid token**

**Word is:nakI**

**Type is: WORD2**

**Word is:cry**

**Lexical error: cry is not a valid token**

**Word is:ikI**

**Type is: WORD2**

**Word is:go\***

**Lexical error: go\* is not a valid token**

**Word is:tabE**

**Type is: WORD2**

**Word is:eat**

**Lexical error: eat is not a valid token**

**Word is:ukE**

**Type is: WORD2**

**Word is:take\***

**Lexical error: take\* is not a valid token**

**Word is:kakI**

**Type is: WORD2**

**Word is:write**

**Lexical error: write is not a valid token**

**Word is:yomI**

**Type is: WORD2**

**Word is:read**

**Lexical error: read is not a valid token**

**Word is:nomI**

**Type is: WORD2**

**Word is:drink**

**Lexical error: drink is not a valid token**

**Word is:agE**

**Type is: WORD2**

**Word is:give**

**Lexical error: give is not a valid token**

**Word is:moraI**

**Type is: WORD2**

**Word is:receive**

**Lexical error: receive is not a valid token**

**Word is:butsI**

**Type is: WORD2**

**Word is:hit**

**Lexical error: hit is not a valid token**

**Word is:kerI**

**Type is: WORD2**

**Word is:kick**

**Lexical error: kick is not a valid token**

**Word is:shaberI**

**Type is: WORD2**

**Word is:talk**

**Lexical error: talk is not a valid token**

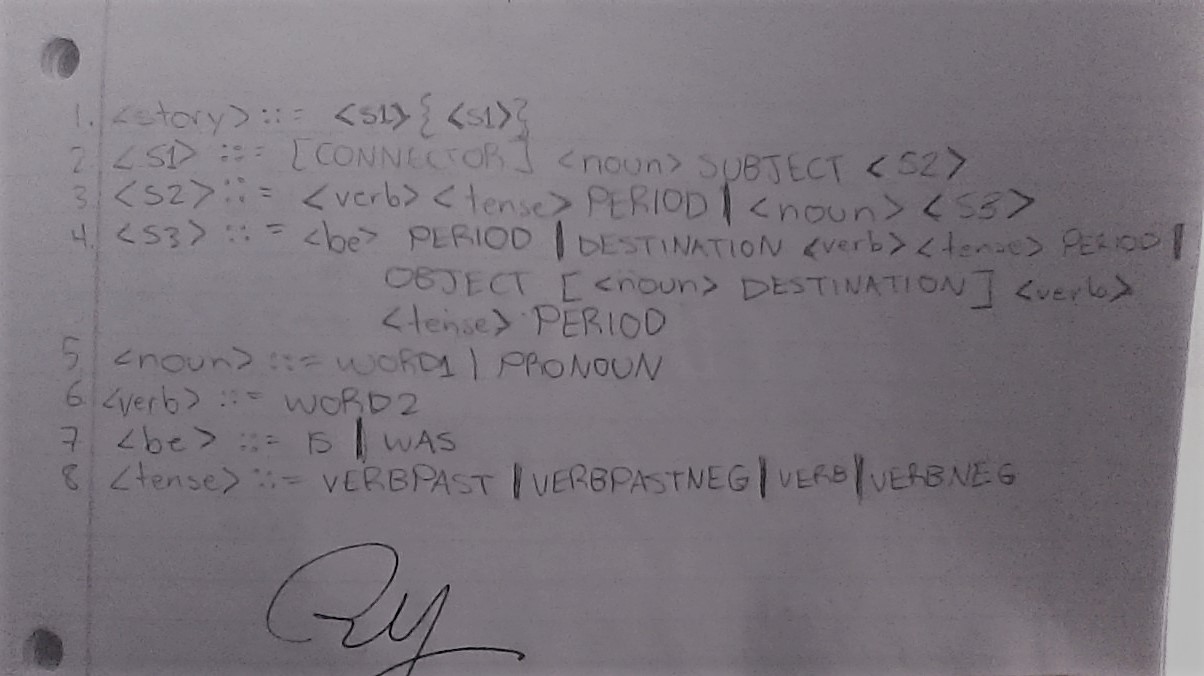
**Word is:eofm**

**Lexical error: eofm is not a valid token**

**]0;marti545@empress:~/cs421/CS421Progs/ScannerFiles[marti545@empress ScannerFiles]$ exit**

**exit**

**Script done on Wed 06 Dec 2017 03:59:09 PM PST**

****

**#include<iostream>**

**#include<fstream>**

**#include<string>**

**#include<cstdlib>**

**#include<vector>**

**#include "scanner.cpp"**

**using namespace std;**

**//=================================================**

**// File parser.cpp written by Group Number: \*\***

**//=================================================**

**/\***

**These will be used to easily print the actual token type name.**

**If tokentype expected is WORD1, or it's actual value 0,**

**then**

**cout << token\_type[expected];**

**will output the string: WORD1**

**\*/**

**string token\_type[] = {"WORD1", "WORD2", "PERIOD", "ERROR", "VERB", "VERBNEG", "VERBPAST", "VERBPASTNEG", "IS", "WAS", "OBJECT", "SUBJECT", "DESTINATION", "PRONOUN", "CONNECTOR", "EOFM"};**

**void noun();**

**void verb();**

**void be();**

**void tense();**

**void afterNOUN();**

**void afterSUBJECT();**

**void s1();**

**string saved\_lexeme; //save the latest read string from file**

**tokentype saved\_token; //save the latest read token type**

**string thefile; //name of file being read**

**bool token\_available = false; //start token\_available as false**

**bool traceON; //checks whether trace should be on or off**

**ofstream fout; //so we can write to file from error functions**

**// \*\* Be sure to put the name of the programmer above each function**

**// i.e. Done by:**

**// \*\* Need syntaxerror1 and syntaxerror2 functions (each takes 2 args)**

**// \*\* Done by: Efren**

**void syntax\_error1(tokentype expected, string lexeme)**

**{**

**cout << "SYNTAX ERROR: Expected " << token\_type[expected] << " but found " << lexeme << "\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "SYNTAX ERROR: Expected " << token\_type[expected] << " but found " << lexeme << "\n\n"; //write to error file**

**exit(1);**

**}**

**void syntax\_error2(string lexeme, string parser\_function)**

**{**

**cout << "SYNTAX ERROR: Unexpected " << lexeme << " found in " << parser\_function << "\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "SYNTAX ERROR: Unexpected " << lexeme << " found in " << parser\_function << "\n\n"; //write to error file**

**exit(1);**

**}**

**// \*\* Need the updated match and next\_token (with 2 global vars)**

**// \*\* Done by: Chris**

**tokentype next\_token()**

**{**

**string lexeme;**

**if(!token\_available)// if there is no saved token from previous lookahead**

**{**

**scanner(saved\_token, lexeme);// call scanner to grab a new token**

**token\_available = true; // mark that fact that you have saved it**

**saved\_lexeme = lexeme; //update saved\_lexeme with latest string from file**

**if(traceON)**

**cout << "Scanner was called..." << endl;**

**if(saved\_token == ERROR) //if the string is an ERROR, then exit**

**{**

**cout << "LEXICAL ERROR: " << lexeme << " is not in the language.\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "LEXICAL ERROR: " << lexeme << " is not in the language.\n\n";**

**exit(1);**

**}**

**}**

**return saved\_token; // return the saved token**

**}**

**bool match(tokentype expected)**

**{**

**if(next\_token() != expected)//mismatch has occurred with the next token**

**{**

**// generate a syntax error message here**

**// do error handling here if any**

**syntax\_error1(expected, saved\_lexeme);**

**}**

**else // match has occurred**

**{**

**token\_available = false; // eat up the token**

**cout << "Matched " << token\_type[expected] << endl; //cout matched token**

**return true; // say there was a match**

**}**

**}**

**// \*\* Make each non-terminal into a function here**

**// \*\* Be sure to put the corresponding grammar rule above each function**

**// \*\* Done by: Chris, Jason, Efren**

**//<story>::= <s1>{<s1>}**

**void story()**

**{**

**while(true)**

**{**

**switch(next\_token())**

**{**

**case EOFM:**

**return;**

**default:**

**cout << endl;**

**if(traceON) //if trace is off, skip these cout's**

**cout << "========== Processing <story> ==========" << endl;**

**s1();**

**break;**

**}**

**}**

**}**

**//<s1>::= [CONNECTOR]<noun>SUBJECT<afterSUBJECT>**

**void s1()**

**{**

**if(traceON)**

**cout << "Processing <s1>" << endl;**

**if(next\_token() == CONNECTOR)**

**match(CONNECTOR);**

**noun();**

**if(next\_token() == SUBJECT)**

**match(SUBJECT);**

**else**

**syntax\_error1(SUBJECT, saved\_lexeme);**

**afterSUBJECT();**

**}**

**//<afterSUBJECT>::= <verb><tense>PERIOD | <noun><afterNOUN>**

**void afterSUBJECT()**

**{**

**if(traceON)**

**cout << "Processing <afterSUBJECT>" << endl;**

**switch(next\_token())**

**{**

**case WORD2:**

**verb();**

**tense();**

**if(next\_token() == PERIOD)**

**match(PERIOD);**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case WORD1:**

**case PRONOUN:**

**noun();**

**afterNOUN();**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "afterSUBJECT");**

**}**

**}**

**//<afterNOUN>::= <be>PERIOD | DESTINATION<verb><tense>PERIOD | OBJECT[<noun>DESTINATION]<verb><tense>PERIOD**

**void afterNOUN()**

**{**

**if(traceON)**

**cout << "Processing <afterNOUN>" << endl;**

**switch(next\_token())**

**{**

**case IS:**

**case WAS:**

**be();**

**if(next\_token() == PERIOD)**

**match(PERIOD);**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case DESTINATION:**

**match(DESTINATION);**

**verb();**

**tense();**

**if(next\_token() == PERIOD)**

**match(PERIOD);**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case OBJECT:**

**match(OBJECT);**

**if(next\_token() == WORD1 || next\_token() == PRONOUN)**

**{**

**noun();**

**if(next\_token() == DESTINATION)**

**match(DESTINATION);**

**else**

**syntax\_error1(DESTINATION, saved\_lexeme);**

**}**

**verb();**

**tense();**

**if(next\_token() == PERIOD)**

**match(PERIOD);**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "afterNOUN");**

**}**

**}**

**//<noun>::= WORD1 | PRONOUN**

**void noun()**

**{**

**if(traceON)**

**cout << "Processing <noun>" << endl;**

**switch(next\_token())**

**{**

**case WORD1:**

**match(WORD1);**

**break;**

**case PRONOUN:**

**match(PRONOUN);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "noun");**

**}**

**}**

**//<verb>::= WORD2**

**void verb()**

**{**

**if(traceON)**

**cout << "Processing <verb>" << endl;**

**switch(next\_token())**

**{**

**case WORD2:**

**match(WORD2);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "verb");**

**}**

**}**

**//<be>::= IS | WAS**

**void be()**

**{**

**if(traceON)**

**cout << "Processing <be>" << endl;**

**switch(next\_token())**

**{**

**case IS:**

**match(IS);**

**break;**

**case WAS:**

**match(WAS);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "be");**

**}**

**}**

**//<tense>::= VERBPAST | VERBPASTNEG | VERB | VERBNEG**

**void tense()**

**{**

**if(traceON)**

**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(saved\_lexeme, "tense");**

**}**

**}**

**// The test driver to start the parser**

**// Done by: Jason**

**int main()**

**{**

**//- opens the input file**

**//- calls the <story> to start parsing**

**//- closes the input file**

**// string thefile;**

**char traceDecision;**

**cout << "Please enter the name of the file: " << endl;**

**cin >> thefile;**

**//loop until a proper answer is given**

**while(traceDecision != 'y' && traceDecision != 'n')**

**{**

**cout << "Do you want trace on? (y or n): ";**

**cin >> traceDecision;**

**if(traceDecision == 'y')**

**traceON = true;**

**else if(traceDecision == 'n')**

**traceON = false;**

**}**

**fin.open(thefile.c\_str());**

**/\***

**Open file to write errors.**

**New errors will be added to the end of the file**

**instead of overwriting what was previously written.**

**\*/**

**fout.open("errors.txt", ios::out | ios::app);**

**story(); //call story() to start parsing**

**fin.close();**

**fout.close();**

**cout << "\n\n";**

**return 0;**

**}// end**

**//\*\* should require no other input files!**

**Script started on Wed 06 Dec 2017 04:01:47 PM PST**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest1**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**Matched IS**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**Matched IS**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched WORD1**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Matched OBJECT**

**Scanner was called...**

**Processing <verb>**

**Matched WORD2**

**Processing <tense>**

**Scanner was called...**

**Matched VERB**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Matched OBJECT**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Scanner was called...**

**Matched DESTINATION**

**Processing <verb>**

**Scanner was called...**

**Matched WORD2**

**Processing <tense>**

**Scanner was called...**

**Matched VERBPAST**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Matched CONNECTOR**

**Processing <noun>**

**Scanner was called...**

**Matched WORD1**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <verb>**

**Matched WORD2**

**Processing <tense>**

**Scanner was called...**

**Matched VERBPASTNEG**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Matched CONNECTOR**

**Processing <noun>**

**Scanner was called...**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**Matched WAS**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Matched CONNECTOR**

**Processing <noun>**

**Scanner was called...**

**Matched WORD1**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Matched DESTINATION**

**Processing <verb>**

**Scanner was called...**

**Matched WORD2**

**Processing <tense>**

**Scanner was called...**

**Matched VERBPAST**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched WORD1**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <verb>**

**Matched WORD2**

**Processing <tense>**

**Scanner was called...**

**Matched VERBPAST**

**Scanner was called...**

**Matched PERIOD**

**Scanner was called...**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest2**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Matched CONNECTOR**

**Processing <noun>**

**Scanner was called...**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**Matched IS**

**Scanner was called...**

**SYNTAX ERROR: Expected PERIOD but found ne**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest3**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Matched CONNECTOR**

**Processing <noun>**

**Scanner was called...**

**Matched PRONOUN**

**Scanner was called...**

**SYNTAX ERROR: Expected SUBJECT but found de**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest4**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**Matched PRONOUN**

**Scanner was called...**

**Matched SUBJECT**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**Matched WORD1**

**Processing <afterNOUN>**

**Scanner was called...**

**SYNTAX ERROR: Unexpected mashita found in afterNOUN**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest5**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**SYNTAX ERROR: Unexpected wa found in noun**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.out**

**Please enter the name of the file:**

**partBtest6**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**LEXICAL ERROR: apple is not in the language.**

**]0;marti545@empress:~/cs421/CS421Progs/ParserFiles[marti545@empress ParserFiles]$ ./group4parser.o[Kexit**

**exit**

**Script done on Wed 06 Dec 2017 04:02:56 PM PST**

**#include<iostream>**

**#include<iomanip>**

**#include<fstream>**

**#include<string>**

**#include<cstdlib>**

**#include<vector>**

**#include "scanner.cpp"**

**using namespace std;**

**//COLOR CODES**

**#define RESET "\033[0m"**

**#define CYAN "\033[36m" /\* Cyan \*/**

**#define GREEN "\033[32m" /\* Green \*/**

**//=================================================**

**// File parser.cpp written by Group Number: \*\***

**//=================================================**

**/\***

**These will be used to easily print the actual token type name.**

**If tokentype expected is WORD1, or it's actual value 0,**

**then**

**cout << token\_type[expected];**

**will output the string: WORD1**

**\*/**

**string token\_type[] = {"WORD1", "WORD2", "PERIOD", "ERROR", "VERB", "VERBNEG", "VERBPAST", "VERBPASTNEG", "IS", "WAS", "OBJECT", "SUBJECT", "DESTINATION", "PRONOUN", "CONNECTOR", "EOFM"};**

**void noun();**

**void verb();**

**void be();**

**void tense();**

**void afterNOUN();**

**void afterSUBJECT();**

**void s1();**

**string saved\_lexeme; //save the latest read string from file**

**tokentype saved\_token; //save the latest read token type**

**string saved\_E\_word; //saved english word for part C**

**string thefile; //name of file being read**

**bool token\_available = false; //start token\_available as false**

**bool traceON; //checks whether trace should be on or off**

**ofstream fout; //so we can write to file from error functions**

**ofstream trs; //write to translated.txt file**

**// \*\* Be sure to put the name of the programmer above each function**

**// i.e. Done by:**

**// Calls lexicon() and retrieves english word, if there is one, and saves it to saved\_E\_word**

**// If there is no english word, then the saved\_lexeme is saved to saved\_E\_word**

**// Done by: Chris**

**void getEword()**

**{**

**lexicon(saved\_lexeme, saved\_E\_word);**

**// cout << "English: " << saved\_E\_word << endl;**

**}**

**//output onscreen and to translated.txt file**

**//Done by: Jason**

**void gen(string name, int typeOf)**

**{**

**switch(typeOf)**

**{**

**case 1:**

**cout << CYAN << name << ": " << saved\_E\_word << RESET << endl;**

**trs << name << ": " << saved\_E\_word << endl;**

**break;**

**case 2:**

**cout << CYAN << name << ": " << token\_type[saved\_token] << RESET << endl;**

**trs << name << ": " << token\_type[saved\_token] << endl;**

**break;**

**case 3:**

**cout << CYAN << name << ": " << saved\_E\_word << endl;**

**cout << CYAN << "TENSE" << ": " << token\_type[saved\_token] << RESET << endl;**

**trs << name << ": " << saved\_E\_word << endl;**

**trs << "TENSE: " << token\_type[saved\_token] << endl;**

**break;**

**}**

**}**

**// \*\* Need syntaxerror1 and syntaxerror2 functions (each takes 2 args)**

**// \*\* Done by: Efren**

**void syntax\_error1(tokentype expected, string lexeme)**

**{**

**cout << "SYNTAX ERROR: Expected " << token\_type[expected] << " but found " << lexeme << "\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "SYNTAX ERROR: Expected " << token\_type[expected] << " but found " << lexeme << "\n\n"; //write to error file**

**exit(1);**

**}**

**void syntax\_error2(string lexeme, string parser\_function)**

**{**

**cout << "SYNTAX ERROR: Unexpected " << lexeme << " found in " << parser\_function << "\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "SYNTAX ERROR: Unexpected " << lexeme << " found in " << parser\_function << "\n\n"; //write to error file**

**exit(1);**

**}**

**// \*\* Need the updated match and next\_token (with 2 global vars)**

**// \*\* Done by: Chris**

**tokentype next\_token()**

**{**

**string lexeme;**

**if(!token\_available)// if there is no saved token from previous lookahead**

**{**

**scanner(saved\_token, lexeme);// call scanner to grab a new token**

**token\_available = true; // mark that fact that you have saved it**

**saved\_lexeme = lexeme; //update saved\_lexeme with latest string from file**

**if(traceON)**

**cout << "Scanner was called..." << endl;**

**if(saved\_token == ERROR) //if the string is an ERROR, then exit**

**{**

**cout << "LEXICAL ERROR: " << lexeme << " is not in the language.\n\n\n";**

**fout << "In file " << thefile << ":" << endl; //write the name of the file opened to the error file**

**fout << "LEXICAL ERROR: " << lexeme << " is not in the language.\n\n";**

**// exit(1);**

**}**

**}**

**return saved\_token; // return the saved token**

**}**

**bool match(tokentype expected)**

**{**

**if(next\_token() != expected)//mismatch has occurred with the next token**

**{**

**// generate a syntax error message here**

**// do error handling here if any**

**syntax\_error1(expected, saved\_lexeme);**

**}**

**else // match has occurred**

**{**

**token\_available = false; // eat up the token**

**cout << GREEN << "Matched " << token\_type[expected] << RESET << endl; //cout matched token**

**return true; // say there was a match**

**}**

**}**

**// \*\* Make each non-terminal into a function here**

**// \*\* Be sure to put the corresponding grammar rule above each function**

**// \*\* Done by: Chris, Jason, Efren**

**// \*\* Updated grammar rules done by: Efren**

**//<story>::= <s1>{<s1>}**

**void story()**

**{**

**while(true)**

**{**

**switch(next\_token())**

**{**

**case EOFM:**

**return;**

**default:**

**cout << endl;**

**if(traceON) //if trace is off, skip these cout's**

**cout << "========== Processing <story> ==========" << endl;**

**s1();**

**break;**

**}**

**}**

**}**

**//<s1>::= [CONNECTOR#getEword##gen#]<noun>#getEword#SUBJECT#gen#<afterSUBJECT>**

**void s1()**

**{**

**if(traceON)**

**cout << "Processing <s1>" << endl;**

**if(next\_token() == CONNECTOR)**

**{**

**match(CONNECTOR);**

**getEword();**

**gen("CONNECTOR", 1);**

**}**

**noun();**

**getEword();**

**if(next\_token() == SUBJECT)**

**{**

**match(SUBJECT);**

**gen("ACTOR", 1);**

**}**

**else**

**syntax\_error1(SUBJECT, saved\_lexeme);**

**afterSUBJECT();**

**}**

**//<afterSUBJECT>::= <verb>#getEword##gen#<tense>#gen#PERIOD | <noun>#getEword#<afterNOUN>**

**void afterSUBJECT()**

**{**

**if(traceON)**

**cout << "Processing <afterSUBJECT>" << endl;**

**switch(next\_token())**

**{**

**case WORD2:**

**verb();**

**getEword();**

**gen("ACTION", 1);**

**tense();**

**gen("TENSE", 2);**

**if(next\_token() == PERIOD)**

**{**

**match(PERIOD);**

**trs << endl;**

**}**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case WORD1:**

**case PRONOUN:**

**noun();**

**getEword();**

**afterNOUN();**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "afterSUBJECT");**

**}**

**}**

**//<afterNOUN>::= <be>#gen#PERIOD | DESTINATION#gen#<verb>#getEword##gen#<tense>#gen#PERIOD | OBJECT#gen#[<noun>#getEword#DESTINATION#gen#]<verb>#getEword##gen#<tense>#gen#PERIOD**

**void afterNOUN()**

**{**

**if(traceON)**

**cout << "Processing <afterNOUN>" << endl;**

**switch(next\_token())**

**{**

**case IS:**

**case WAS:**

**be();**

**gen("DESCRIPTION", 3);**

**if(next\_token() == PERIOD)**

**{**

**match(PERIOD);**

**trs << endl;**

**}**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case DESTINATION:**

**match(DESTINATION);**

**gen("TO", 1);**

**verb();**

**getEword();**

**gen("ACTION", 1);**

**tense();**

**gen("TENSE", 2);**

**if(next\_token() == PERIOD)**

**{**

**match(PERIOD);**

**trs << endl;**

**}**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**case OBJECT:**

**match(OBJECT);**

**gen("OBJECT", 1);**

**if(next\_token() == WORD1 || next\_token() == PRONOUN)**

**{**

**noun();**

**getEword();**

**if(next\_token() == DESTINATION)**

**{**

**match(DESTINATION);**

**gen("TO", 1);**

**}**

**else**

**syntax\_error1(DESTINATION, saved\_lexeme);**

**}**

**verb();**

**getEword();**

**gen("ACTION", 1);**

**tense();**

**gen("TENSE", 2);**

**if(next\_token() == PERIOD)**

**{**

**match(PERIOD);**

**trs << endl;**

**}**

**else**

**syntax\_error1(PERIOD, saved\_lexeme);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "afterNOUN");**

**}**

**}**

**//<noun>::= WORD1 | PRONOUN**

**void noun()**

**{**

**if(traceON)**

**cout << "Processing <noun>" << endl;**

**switch(next\_token())**

**{**

**case WORD1:**

**match(WORD1);**

**break;**

**case PRONOUN:**

**match(PRONOUN);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "noun");**

**}**

**}**

**//<verb>::= WORD2**

**void verb()**

**{**

**if(traceON)**

**cout << "Processing <verb>" << endl;**

**switch(next\_token())**

**{**

**case WORD2:**

**match(WORD2);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "verb");**

**}**

**}**

**//<be>::= IS | WAS**

**void be()**

**{**

**if(traceON)**

**cout << "Processing <be>" << endl;**

**switch(next\_token())**

**{**

**case IS:**

**match(IS);**

**break;**

**case WAS:**

**match(WAS);**

**break;**

**default:**

**syntax\_error2(saved\_lexeme, "be");**

**}**

**}**

**//<tense>::= VERBPAST | VERBPASTNEG | VERB | VERBNEG**

**void tense()**

**{**

**if(traceON)**

**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(saved\_lexeme, "tense");**

**}**

**}**

**// The test driver to start the parser**

**// Done by: Jason**

**int main()**

**{**

**//- opens the input file**

**//- calls the <story> to start parsing**

**//- closes the input file**

**// string thefile;**

**char traceDecision;**

**cout << "Please enter the name of the file: " << endl;**

**cin >> thefile;**

**//loop until a proper answer is given**

**while(traceDecision != 'y' && traceDecision != 'n')**

**{**

**cout << "Do you want trace on? (y or n): ";**

**cin >> traceDecision;**

**if(traceDecision == 'y')**

**traceON = true;**

**else if(traceDecision == 'n')**

**traceON = false;**

**}**

**fin.open(thefile.c\_str());**

**/\***

**Open file to write errors.**

**New errors will be added to the end of the file**

**instead of overwriting what was previously written.**

**\*/**

**fout.open("errors.txt", ios::out | ios::app); //write errors to**

**trs.open("translated.txt"); //open file to write translated lines to**

**story(); //call story() to start parsing**

**fin.close();**

**fout.close();**

**trs.close();**

**cout << "\n\n";**

**return 0;**

**}// end**

**//\*\* should require no other input files!**

**Script started on Thu 07 Dec 2017 02:14:59 PM PST**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest1**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**[32mMatched IS[0m**

**[36mDESCRIPTION: rika**

**[36mTENSE: IS[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**[32mMatched IS[0m**

**[36mDESCRIPTION: sensei**

**[36mTENSE: IS[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: rika[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**[32mMatched OBJECT[0m**

**[36mOBJECT: gohan[0m**

**Scanner was called...**

**Processing <verb>**

**[32mMatched WORD2[0m**

**[36mACTION: tabE[0m**

**Processing <tense>**

**Scanner was called...**

**[32mMatched VERB[0m**

**[36mTENSE: VERB[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**[32mMatched OBJECT[0m**

**[36mOBJECT: tesuto[0m**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Scanner was called...**

**[32mMatched DESTINATION[0m**

**[36mTO: seito[0m**

**Processing <verb>**

**Scanner was called...**

**[32mMatched WORD2[0m**

**[36mACTION: agE[0m**

**Processing <tense>**

**Scanner was called...**

**[32mMatched VERBPAST[0m**

**[36mTENSE: VERBPAST[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**[32mMatched CONNECTOR[0m**

**[36mCONNECTOR: However[0m**

**Processing <noun>**

**Scanner was called...**

**[32mMatched WORD1[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: seito[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <verb>**

**[32mMatched WORD2[0m**

**[36mACTION: yorokobI[0m**

**Processing <tense>**

**Scanner was called...**

**[32mMatched VERBPASTNEG[0m**

**[36mTENSE: VERBPASTNEG[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**[32mMatched CONNECTOR[0m**

**[36mCONNECTOR: Therefore[0m**

**Processing <noun>**

**Scanner was called...**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**[32mMatched WAS[0m**

**[36mDESCRIPTION: kanashii**

**[36mTENSE: WAS[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**[32mMatched CONNECTOR[0m**

**[36mCONNECTOR: Then[0m**

**Processing <noun>**

**Scanner was called...**

**[32mMatched WORD1[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: rika[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**[32mMatched DESTINATION[0m**

**[36mTO: toire[0m**

**Processing <verb>**

**Scanner was called...**

**[32mMatched WORD2[0m**

**[36mACTION: ikI[0m**

**Processing <tense>**

**Scanner was called...**

**[32mMatched VERBPAST[0m**

**[36mTENSE: VERBPAST[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: rika[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <verb>**

**[32mMatched WORD2[0m**

**[36mACTION: nakI[0m**

**Processing <tense>**

**Scanner was called...**

**[32mMatched VERBPAST[0m**

**[36mTENSE: VERBPAST[0m**

**Scanner was called...**

**[32mMatched PERIOD[0m**

**Scanner was called...**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest2**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**[32mMatched CONNECTOR[0m**

**[36mCONNECTOR: Then[0m**

**Processing <noun>**

**Scanner was called...**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**Processing <be>**

**[32mMatched IS[0m**

**[36mDESCRIPTION: rika**

**[36mTENSE: IS[0m**

**Scanner was called...**

**SYNTAX ERROR: Expected PERIOD but found ne**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest3**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**[32mMatched CONNECTOR[0m**

**[36mCONNECTOR: Therefore[0m**

**Processing <noun>**

**Scanner was called...**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**SYNTAX ERROR: Expected SUBJECT but found de**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest4**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**[32mMatched PRONOUN[0m**

**Scanner was called...**

**[32mMatched SUBJECT[0m**

**[36mACTOR: I/me[0m**

**Processing <afterSUBJECT>**

**Scanner was called...**

**Processing <noun>**

**[32mMatched WORD1[0m**

**Processing <afterNOUN>**

**Scanner was called...**

**SYNTAX ERROR: Unexpected mashita found in afterNOUN**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest5**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**SYNTAX ERROR: Unexpected wa found in noun**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ ./group4.out**

**Please enter the name of the file:**

**partBtest6**

**Do you want trace on? (y or n): y**

**Scanner was called...**

**LEXICAL ERROR: apple is not in the language.**

**========== Processing <story> ==========**

**Processing <s1>**

**Processing <noun>**

**SYNTAX ERROR: Unexpected apple found in noun**

**]0;marti545@empress:~/cs421/CS421Progs/TranslatorFiles[marti545@empress TranslatorFiles]$ exit**

**exit**

**Script done on Thu 07 Dec 2017 02:16:08 PM PST**