Course: CSC 340.03
Student: Aisha Asif, SFSU ID: 920560260
Teammate: Michael Chang, SFSU ID: 920899275
Assignment Number: 03
Assignment Due Date & Time: 4/17/22 at 11:59pm

Assignment Report

Part A: Tic - Tac - Toe

```
 | |
-----------
 | |
-----------
 | |
Enter a row (0, 1, 2) for player X : 0
Enter a column (0, 1, 2) for player X : 0
X| |
-----------
 | |
-----------
 | |
Enter a row (0, 1, 2) for player O : 1
Enter a column (0, 1, 2) for player O : 1
X| |
-----------
 |O|
-----------
 | |
Enter a row (0, 1, 2) for player X : 0
Enter a column (0, 1, 2) for player X : 1
X|X|
-----------
 |O|
-----------
 | |
Enter a row (0, 1, 2) for player O : 2
Enter a column (0, 1, 2) for player O : 2
X|X|
-----------
 |O|
-----------
 | |O
Enter a row (0, 1, 2) for player X : 0
Enter a column (0, 1, 2) for player X : 2
X|X|X
-----------
 |O|
-----------
 | |O
X player won
```

```
 | |
-----------
 | |
-----------
 | |
Enter a row (0, 1, 2) for player X : 0
Enter a column (0, 1, 2) for player X : 0
X| |
-----------
 | |
-----------
 | |
Enter a row (0, 1, 2) for player O : 0
Enter a column (0, 1, 2) for player O : 0
This position is already occupied. Please enter a new position
Enter a row (0, 1, 2) for player O :
```

```
 | |
-----------
 | |
-----------
 | |
Enter a row (0, 1, 2) for player X : 1
Enter a column (0, 1, 2) for player X : 1
 | |
-----------
 |X|
-----------
 | |
Enter a row (0, 1, 2) for player O : 2
Enter a column (0, 1, 2) for player O : 2
 | |
-----------
 |X|
-----------
 | |O
Enter a row (0, 1, 2) for player X : 1
Enter a column (0, 1, 2) for player X : 3
Out of Range try again
```

Assignment-03_PA_TicTacToe.cpp > main()

```cpp
/*
*
* File: TicTacToe.cpp
* By: Aisha Asif
* Date: 05/14/2022
*
* Description: this code is the tictactoe game!
*
*/
#include <iostream>
using namespace std;

bool isWon(char, char[][3]);
bool isDraw(char[][3]);
void displayBoard(char[][3]);
void makeAMove(char[][3], char);
int const maximumMoves(9);
int numberOfMoves(0);

int main() {
    //
    //  PLEASE DO NOT CHANGE function main
    //
    char board[3][3] = { { ' ', ' ', ' ' },{ ' ', ' ', ' ' },{ ' ', ' ', ' ' } };
    displayBoard(board);

    while (true) {

        // The first player makes a move
        makeAMove(board, 'X');
        displayBoard(board);

        if (isWon('X', board)) {
            cout << "X player won" << endl;
            exit(0);
        }
        else if (isDraw(board)) {
            cout << "No winner" << endl;
            exit(0);
        }
```

Assignment-03_PA_TicTacToe.cpp > main()

```
        // The second player makes a move
        makeAMove(board, 'O');
        displayBoard(board);

        if (isWon('O', board)) {
            cout << "O player won" << endl;
            exit(0);
        }
        else if (isDraw(board)) {
            cout << "No winner" << endl;
            exit(0);
        }
    }

    return 0;
}
// for winning the game
bool isWon (char turn, char gamer [] [3]){
    if (gamer [0][0] == turn && gamer[0][1] == turn && gamer[0][2] == turn){
        return true;
    }else if (gamer[1][0] == turn && gamer [1][1] == turn && gamer [1][2] == turn){
        return true;
    }else if (gamer[2][0] == turn && gamer [2][1] == turn && gamer [2][2] == turn){
        return true;
    }else if (gamer[0][0] == turn && gamer [1][0] == turn && gamer [2][0] == turn){
        return true;
    }else if (gamer[0][1] == turn && gamer [1][1] == turn && gamer [2][1] == turn){
        return true;
    }else if (gamer[0][2] == turn && gamer [1][2] == turn && gamer [2][2] == turn){
        return true;
    }else if (gamer[0][0] == turn && gamer [1][1] == turn && gamer [2][2] == turn){
        return true;
    }else if (gamer[2][0] == turn && gamer [1][1] == turn && gamer [0][2] == turn){
        return true;
    }return false;
//when the game is at a draw (when the number of moves is equal to the maximum moves and no one has w
}bool isDraw(char gamer [][3]){
    if (numberOfMoves == maximumMoves){
        return true;
    }
```

Assignment-03_PA_TicTacToe.cpp > main()

```
76      }return false;
77  //when the game is at a draw (when the number of moves is equal to the maximum moves and no one has w
78  }bool isDraw(char gamer [][3]){
79      if (numberOfMoves == maximumMoves){
80          return true;
81      }
82      return false;
83  //the tic tac toe board printed out in a series of arrays and cout hard-coded
84  }void displayBoard(char gamer[][3]){
85          cout << gamer[0][0]<< "|"<< gamer[0][1]<<"|" <<gamer[0][2]<< endl;
86          cout << "------------" << endl;
87          cout << gamer[1][0]<< "|"<< gamer[1][1]<<"|" <<gamer[1][2]<< endl;
88          cout << "------------" << endl;
89          cout << gamer[2][0]<< "|"<< gamer[2][1]<<"|" <<gamer[2][2]<< endl;
90  //if the gamer makes a move and it is not one of the options //voided
91  }void makeAMove(char gamer[][3], char turn) {
92     ++numberOfMoves;
93      int row;
94      int column;
95      bool placement = false;
96      while(placement == false) {
97          cout << "Enter a row (0, 1, 2) for player " << turn << " : ";
98          cin >> row;
99          cout << "Enter a column (0, 1, 2) for player " << turn  << " : ";
100         cin >> column;
101         if(row < 0 || row > 2 || column < 0 || column > 2) {
102             cout << "Out of Range try again" << endl;
103     //gamer make a move. if the move is in the same spot as another move it is void
104         }else {
105             if(gamer[row][column] == ' ') {
106                 gamer[row][column] = turn;
107                 break;
108             }else {
109                 cout << "This position is already occupied. Please enter a new position" << endl;
110             }
111         }
112     }
113
114
115 }
```

Part B: Credit Card

Assignment-03_PB_CCNumberValidation.cpp > main()

```cpp
/*
 *
 * File: Dictionary.cpp
 * By: Aisha Asif & Michael Chang
 * Date: 05/14/2022
 *
 * Description: this code allows the user to use a credit card!
 *
 */

#include<iostream>
#include<string>
#include<vector>
#include<iomanip>
using namespace std;

bool isvalidcc(const string&);

int main()
{
    //
    // PLEASE DO NOT CHANGE function main
    //
    vector<string> cardnumbers = {
        "371449635398431", "4444444444444448", "4444424444444440", "4110144110144115",
        "4114360123456785", "4061724061724061", "5500005555555559", "5115915115915118",
        "5555555555555557", "6011016011016011", "372449635398431", "4444544444444448",
        "4444434444444440", "4110145110144115", "4124360123456785", "4062724061724061",
        "5501005555555559", "5125915115915118", "5556555555555557", "6011116011016011",
        "372449635397431", "4444544444444448", "4444434444544440", "4110145110184115",
        "4124360123457785", "4062724061724061", "5541005555555559", "5125115115915118",
        "5556551555555557", "6011316011016011"
    };

    int i;
    vector<string>::iterator itr;

    for (i = 1, itr = cardnumbers.begin(); itr != cardnumbers.end(); ++itr, i++) {
        cout << setw(2)  << i << " "
             << setw(17) << *itr
             << ((isvalidcc(*itr)) ? " is valid" : " is not valid\n");
```

Assignment-03_PB_CCNumberValidation.cpp > main()

```cpp
    return 0;
}

//this will go through all of the list of vectors and look for values that are 4, 5, 6, 37

bool isvalidcc(const string& string){
    int oneDigit;
    int oddDigit;
    int total(0);
    if (string.length() < 13 || string.length() > 16){}
    if(!(string [0] == '4' || string [0] == '5' || string [0] == '6' || string.substr (0, 2) == "37"))
    {  return false; }
oneDigit = doubleDigit(string);
oddDigit = unevenDigit(string);
total = oddDigit + oneDigit;

if(total%10 != 0)
{ return false;}
return true;
}

int doubleDigit (string value){
    int total = 0;
    for(int i =value.length() - 1; i >= 0; i -= 2){
        total += value[i] - '0';
    }
return total;
}

int unevenDigit(string value){
    int total = 0;
    for (int i = value.length() - 2; i >= 0; i -= 2){
        int num = (value[i] - '0') * 2;
        if(num > 9){
            num = num/10 + num%10;
        }
        total += num;
    }
    return total;
}
```

Part C:

```
! Opening data file... ./Data.CS.SFSU.txt
! Loading data...
! Closing data file..../Data.CS.SFSU.txt
====== Libary 340 C++ =====
------ Keywords: 19
------ Definitions: 61

Search [1] :interjection
      |
   Interjection [noun] : Interjection is a short sound, word or phrase spoken suddenly to express an emotion. Oh
!, Look out! and Ow! are interjections.
      |
Search [2] :arrow
      |
   Arrow [noun] : Here is one arrow: <IMG> -=>> </IMG>..
      |
Search [3] :noun noun reverse
      |
      Noun [noun] : Noun is a word that refers to a person, (such as Ann or doctor), a place (such as Paris or c
ity) or a thing, a quality or an activity (such as plant, sorrow or tennis).
      |
Search [4] :book noun distinct reverse
      |
      Book [noun] : A written work published in printed or electronic form.
      Book [noun] : A set of pages.
      |
Search [5] :reverse noun ok distinct
      |
      <The entered 3rd parameter 'ok' is NOT 'distinct'.>
      <The entered 3rd parameter 'ok' is NOT 'reverse'.>
      <The 3rd parameter should be 'distinct' or 'reverse'.>
      <The entered 4th parameter 'distinct' is NOT 'reverse'.>
      <The 4th parameter should be 'reverse'.>
      Reverse [noun] : Change to opposite direction.
      Reverse [noun] : A dictionary program's parameter.
      Reverse [noun] : The opposite.
      Reverse [noun] : To be updated...
      Reverse [noun] : To be updated...
      Reverse [noun] : To be updated...
      Reverse [noun] : To be updated...
      |
```

```
! Loading data...
! Closing data file..../Data.CS.SFSU.txt
====== Libary 340 C++ =====
------ Keywords: 19
------ Definitions: 61

Search [1] :!help
      |
      PARAMETER HOW-TO, please enter:
      1. A search key -then 2. An optional part of speech -then
      3. An optional 'distinct' -then 4. An optional 'reverse'
      |
Search [2] :!q
      |

-----THANK YOU-----
```

```
! Loading data...
! Closing data file..../Data.CS.SFSU.txt
====== Libary 340 C++ =====
------ Keywords: 19
------ Definitions: 61

Search [1] :candyland
      |
      <NOT FOUND> To be considered for the next release. Thank you.
      |
      |
      PARAMETER HOW-TO, please enter:
      1. A search key -then 2. An optional part of speech -then
      3. An optional 'distinct' -then 4. An optional 'reverse'
      |
Search [2] :
```

```cpp
/*
* File: Dictionary.cpp
* By: Aisha Asif & Michael Chang
* Date: 05/14/2022
*
* Description: this code allows the user to search up a word and print out the part of speech and definitions.
*  you are also able to revers how the definiions are printed and search it by the noun.
*
*/

#include<iostream>
#include<string>
#include<fstream>
#include<vector>
#include<algorithm>
#include<sstream>
using namespace std;

bool isanum(string&);
vector<string> textEnty (string&);

class Libary {           // this class allows the user to separet the text file into three sections the word, pos and definistions.
    private:
        string words;
        string speech;
        string definition;

    public:
        Libary(string word, string pos, string def) {
            words = word;
            speech = pos;
            definition = def;
        }
        string getWord() {
            return words;
        }
        string getPos() {
            return speech;
        }
```

```
        }
        string getDef() {
            return definition;
        }
        void setWord(string data) {
            words = data;
        }
        void setPos(string data) {
            speech = data;
        }
        void setDef(string data) {
            definition = data;
        }
        string toString() {              // a tostring tha puts all three parameters in to one string that shows the word, pos and de
            string text = words;
            if(isanum(text) == true) {
                for(auto& data: text) {
                    data = toupper(data);
                }
            }
            text[0] = toupper(text[0]);// this capatalizes the first word of the string
            return text + " [" + speech + "] : " +definition;
        }
        bool operator > (const Libary& str) const {
            return (speech < str.speech);
        }
        bool operator==(const Libary& str) const {
            if(speech != str.speech || definition != str.definition) {
                return false;
            }
            return true;
        }
};

vector<string> Keywordss;
vector<Libary> sort(vector<Libary> lib, vector<string> words);
void search(vector<Libary> lib, vector<string> words);
vector<Libary> part_of_speech(string, vector<Libary>);
```

```cpp
vector<Libary> sort(vector<Libary> lib, vector<string> words);
void search(vector<Libary> lib, vector<string> words);
vector<Libary> part_of_speech(string, vector<Libary>);
vector<Libary> distinct (vector<Libary>);
vector<Libary> reverse (vector<Libary>);
vector<string> pos = {"noun", "verb", "adjective", "adverb", "pronoun", "preposition", "conjunction", "interjection"};

int main(){
    bool opened(true);
    string opened_file("./Data.CS.SFSU.txt");
    ifstream fileHandle(opened_file);
    vector<Libary> lib;
    vector<string> words;

   lib =  sort(lib, words);
   words = Keywordss;
    cout << "! Closing data file..." << opened_file<< endl;
    fileHandle.close();
    cout << "====== Libary 340 C++ ===== " << endl;
    cout << "------ Keywords: " << words.size() << endl; // tells user the number of keywords
    cout << "------ Definitions: " << lib.size() << endl << endl; // tells users the number of definitions

    search(lib, words);

    return 0;
}

bool isanum(string& str) {    /// checks the string for any numbers that were inputed
    return any_of(str.begin(), str.end(), ::isdigit);
}

vector<string> textEnty (string& str) {  // makes the entries made and seperates them
    vector<string> entry;
    string pull;
    stringstream seperate(str);
    while (getline(seperate, pull, ' ')){
        entry.push_back(pull);
    }
```

```
    vector<Libary> pos;
    for(Libary i : data) {
        if(i.getPos() == partsOfSpeech) {
            pos.push_back(i);
        }
    }
    return pos;
}

vector<Libary> distinct (vector<Libary> data) {   // gets rid of the multiples of the same deffintion and only shows one output for t
    vector<Libary> doup = data;
    bool multi(false);
    if(doup.size() > 1){
        doup.erase(unique(doup.begin(), doup.end()), doup.end());

    }
    return doup;
}

vector<Libary> reverse(vector<Libary> data) {     //reverses the order definitions from bottom to top
    vector<Libary> reverse;
    for(int i = data.size() - 1 ; i >= 0; i--) {
        reverse.push_back(data[i]);
    }
    return reverse;
}

vector<Libary> sort(vector<Libary> lib, vector<string> words){
  bool opened(true);
    string opened_file("./Data.CS.SFSU.txt");
    ifstream fileHandle(opened_file);
    vector<Libary> libb = lib;
    vector<string> text = words;
  while( opened){
       ifstream fin(opened_file);
        string line;
        if(fileHandle.is_open()){
            cout << "! Opening data file... " << opened_file << endl;
```

```
            int spliter(line.find_first_of("|"));   // by looking at the | that seperates the word from the nouns and the -=>> th
            int pointer(line.find_first_of("-=>>"));
            string word(line.substr(0, spliter));
            text.push_back(word);
            string pos;
            string def;
            line.erase(0, spliter+1 );
            bool runs = true;
            while(runs){
                spliter = line.find_first_of("|");
                pointer = line.find_first_of("-=>>");
                if(spliter == -1){
                    pos = line.substr(0, pointer - 1);
                    line.erase(0, pointer + 5);
                    def = line;
                    Libary pull(word, pos, def);
                    libb.push_back(pull); // puts all the inputs from the txt into vectors and separates them.
                    break;
                }else{
                    pos = line.substr(0, pointer - 1);
                    line.erase(0, pointer + 5);
                    spliter = line.find_first_of("|");
                    def = line.substr(0, spliter);
                    line.erase(0, spliter + 1);
                    Libary pull(word, pos, def);
                    libb.push_back(pull);

                }
            }

        }
         break;
    }else{
        cout << "<!>ERROR<!> ===> File could not be opened." << endl;
        cout << "<!>ERROR<!> ===> Provided file path:" << opened_file <<endl;
        cout << "<!>Enter the CORRECT data file path:";
        cin >> opened_file;
    }
```

```
  return libb ;
}


void search(vector<Libary> lib, vector<string> words){
    bool runs(true);
    int count(0);

    vector<string> text = words;
    while(runs){
        string user_entry;
        vector<string> entry;
        bool pos(false);
        bool pas(false);
        count++; // increases the number count for the search as more words are searched the count increaeses
        sort(lib.begin(), lib.end(), greater<Libary>());
        cout << "Search [" << count << "] :";

        getline(cin, user_entry);

        for(auto& c: user_entry){
            c = tolower(c);
        }
        entry = textEnty(user_entry);

        cout << "      |" << endl;
        if(entry[0] == "!help" || entry[0] == "" ) {  // helps the user with the params that are need to find specific words
            cout << "      PARAMETER HOW-TO, please enter:" << "\n"
                 << "      1. A search key -then 2. An optional part of speech -then" << "\n"
                 << "      3. An optional 'distinct' -then 4. An optional 'reverse'" << endl;
           pas = true;
        } else if(entry[0] == "!q" || entry[0] == "!Q") {
            cout << endl << "-----THANK YOU-----" << endl;
            break;
        }


        vector<Libary> pull = lib;
```

```
    if(entry.size() == 2){ // call the  parameter weather it be either the part of speech or reverse or distinct
        if(entry[1] == "distinct"){// checks to see if the user inputs distinct as a pram
            pull = distinct(pull);
        }
        if( entry[1] == "reverse"){// checks to see if the user inputs reverse as a pram
            pull = reverse(pull);
        }
        if(find(entry.begin(), entry.end(), entry[1]) != entry.end() && entry[1] != "distinct" && entry[1] != "reverse"){
            pos = true;
            pull = part_of_speech(entry[1], pull);
        }
        if(entry[1] != "distinct" && entry[1] != "reverse" && pos == false){
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT a part of speech.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'distinct'.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'reverse'.>" << endl;
            cout << "      <The 2nd parameter should be a part of speech or 'distinct' or 'reverse'.>" << endl;
        }
        for(Libary i: pull){
            if(i.getWord() == entry[0]){
                cout << "      " << i.toString() << "\n";
            }
        }
    }
    if(entry.size() == 3){  // allows the user to call both the pos and the reverse or the distinct
        if(entry[2] == "distinct" || entry[1] == "distinct"){
            pull = distinct(pull);
        }
        if( entry[2] == "reverse" || entry[1] == "reverse"){
            pull = reverse(pull);
        }
        if(find(entry.begin(), entry.end(), entry[1]) != entry.end() && entry[1] != "distinct" && entry[1] != "reverse"){
            pos = true;
            pull = part_of_speech(entry[1], pull);
        }if(entry[1] != "distinct" && entry[1] != "reverse" && pos == false){
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT a part of speech.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'distinct'.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'reverse'.>" << endl;
            cout << "      <The 2nd parameter should be a part of speech or 'distinct' or 'reverse'.>" << endl;
```

```
                cout << "      " << i.toString() << "\n";
            }
        }
    }
    if(entry.size() == 4 ){ // allows the user to call both the pos and the reverse or the distinct  and the last param as revers
        if(entry[1] == "distinct" || entry[2] == "distinct"){
            pull = distinct(pull);
        }
        if( entry[1] == "reverse" || entry[2] == "reverse" || entry[3] == "reverse"){
            pull = reverse(pull);
        }
        if(find(entry.begin(), entry.end(), entry[1]) != entry.end() && entry[1] != "distinct" && entry[1] != "reverse"){
            pos = true;
            pull = part_of_speech(entry[1], pull);
        }if(entry[1] != "distinct" && entry[1] != "reverse" && pos == false){
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT a part of speech.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'distinct'.>" << endl;
            cout << "      <The entered 2nd parameter " << "'" << entry[1] << "' is NOT 'reverse'.>" << endl;
            cout << "      <The 2nd parameter should be a part of speech or 'distinct' or 'reverse'.>" << endl;
        }if (entry[2] != "distinct" && entry[2] != "reverse"){
           cout << "      <The entered 3rd parameter " << "'" << entry[2] << "' is NOT 'distinct'.>" << endl;
            cout << "      <The entered 3rd parameter " << "'" << entry[2] << "' is NOT 'reverse'.>" << endl;
            cout << "      <The 3rd parameter should be 'distinct' or 'reverse'.> " << endl;
        }if(entry[3] != "reverse"){
            cout << "      <The entered 4th parameter " << "'" << entry[3] << "' is NOT 'reverse'.>" << endl;
            cout << "      <The 4th parameter should be 'reverse'.> " << endl;
        }
        for(Libary i: pull){
            if(i.getWord() == entry[0]){
                cout << "      " << i.toString() << "\n";
            }
        }
    }
    cout << "      |" << endl;
  }
}
```

The code begins with the first ‘isanum’ which basically checks to see if the text is real and turns the text into a pointer which turns it into a calling type which is numbers and letters. If it is true (the text is there) it goes into a for loop to make sure everything is correct. This basically acts like a to-string. The operators are acting as an overload in Java. The operators are greater than and equals equals which turns it into a type of pointer.

All the vectors that are placed in lines 74 - 80 are calling functions that are placed in the main. Some variables include things like ‘keywords’ and ‘pos’. ‘Pos’ is basically the part of speech, ‘keywords’ allow the program to find how many words there are in the dictionary. Inside the main method, the file opener is called to check if the file is there and correct. This is where the text comes from. The ifstream is used to read through the text from the text file. The

vector library goes into the library class and helps get the data (vector string words) which are the keywords required from the dictionary output.

Moving onto “lib == sort(lib, words)” which is calling the parameters of lib and word which are the parameters the program has set for the sort function. The sort function goes through and looks through each line by line in the text file and pulls out if there is a bar that separates between the word and the part of speech. After the “-=>>” is the definition. The word is erased afterward. Once the word is obtained it is pushed all the way into the “library pull(word, pos, def)” which are the parameters for the library. It is then pushed back to the vectors.

The error messages help tell the user if the file can not be opened due to incorrect data file paths and shows what file path was provided. If the file is incorrect the program will help you input the correct file and will continue the program. The library is then returned and keywords are returned as words from the text file (basically the dictionary words from the text file).

The next part is, once the text file is read, there are two parameters in the search (they are the same as the sort) which are library (lib) and words. When looking at the void, we make sure that it runs and if it runs as true (boolean). The count increases each time a new word is counted and searched- same with the definition of the words. The whole dictionary sorts each function from the beginning to the end and secures the definitions.

The next part, ‘getline(cin, user_entry)’ is an easier way to search rather than to always type a line saying to look at the previous entry and switch it. The user entry is taken and made

lowercase so that the program can read it as lowercase only. TextEntry basically allows the inputs to be split like in Java. !help tells the user how the information should be entered and when a user enters something that is not taken by the program, the help prints out the help menu and guides the user on how to use the dictionary. If the word given is in the dictionary, then the word is searched by its parameters in the library. When the user types in !q the user is quit out of the program and a thank you message is printed.

The distinct function basically looks for the same version of the same phrase, it will put anything out and get rid of all of the multiples, and the definition for the part of speech and only use one. If you, for example, searched up “noun” and there were multiple definitions, and the function sees one that has the same definition over and over again it gets rid of that same definition and puts it together, and outputs that information.

Reverse basically reverses the order. The first definition becomes the last definition and the last definition becomes the first definition. So it turns the whole thing around and goes from the last entry to the first entry.

Part of the speech function will basically look through the part of the speech array and print out the part of the speech. If for example, you were looking for a “distinct noun” it will only show the noun parts of speech for that word. One is basically in the array because an array starts at zero. If the noun for example is 2 it will make sure that it looks for the noun at that particular position and print it. The reason the entry doesn’t equal distinct and doesn't equal reverse is that it won’t print out the distinct and reverse if it sees that the entry doesn’t equal the parameters

given. The error messages are further added if you don't put in the correct parameters for the dictionary to read and then you are shown what to do and what you did wrong.

The best way I can think of improving this program is to figure out another algorithm for the code because it takes a while for the code to go through the text file and print out the results. This could be an issue, especially if the text file included more information such as more words, more parts of speech, or more definitions of each word. With just 19 words my computer started breathing like a dragon as I tried to run this code. I would definitely find a way to write a program that did not loop back into the text file and go line by line as often. I would figure out a way so that the program could pull information and display it more efficiently without running so much programs and looping through so many hoops just to get the output the user desires.