Computer Programming Language

【Fall, 2020】

Homework 6

Program A： Bridge Card Game (50%)

In a typical card game, each player gets a hand of cards. The deck is shuffled and cards are dealt one at a time from the deck and added to the players' hands. Bridge uses a pack of 52 playing cards. There are 4 suits (spades, hearts, diamonds, clubs) each of 13 cards: 1 (the Ace) to 10 and Jack, Queen, King. The Ace is the highest card, followed by the King, Queen, down to the 2.

You are requested to design a program to simulate dealing cards to 4 players. The program needs to simulate three times of card shuffling and dealing, and saves the results into a file named **CardGame.txt** as well as displaying the results on the screen. Your program should also calculate and show the total high-card points of each player. High-card points are counted as follows: Ace = 4 points, King = 3 points, Queen = 2 points, and Jack = 1 point. Design your own format for the program output.

■ *AUTOLAB Submission Check:*

```
int answer1;  // Store the total high-card points of the first player in the final run of dealing.
int answer2;  // Store the total high-card points of the second player in the final run of dealing.
int answer3;  // Store the total high-card points of the third player in the final run of dealing.
int answer4;  // Store the total high-card points of the fourth player in the final run of dealing.
```

Program B： Reading Level of a Document (50%)

In this exercise, you are asked to write a program that reads a document in English and calculates the reading level of the document. You will be given a dictionary file (**Dictionary_1000.txt**) containing the top 1000 most commonly used English words based on frequency analysis of a collection of novels. The dictionary file consists of lines of integer rank (1 = most common, 1000 = least common) followed by the word.

To estimate the reading level of a document, you need to read the words from a document file and look up the word in the dictionary file to obtain the rank of each word, and then calculate the average word rank, indicating the reading level, of all words in the document.

For example, consider the first paragraph of "Cat in the Hat" by Dr. Seuss.

The sun did not shine.
It was too wet to play.
So we sat in the house.
All that cold, cold, wet day.

The corresponding ranks for these words with the average rank are:

1 672 77 20 1001
11 8 106 1001 3 617
33 54 273 7 1 133
35 10 463 463 1001 123

The average rank is 265.78.

The average rank, the reading level of the paragraph, of these 23 words is 265.78. Notice that there are two words "shine" and "wet" that are not in the dictionary. For these cases, 1001 is assigned as the rank.

A document file (**Document.txt**) is provided for you to test your program. You need to design you program with a modular approach using functions. You are requested to read the words from the dictionary file and store them in a list of string using dynamic memory allocation approach (pointer or vector (STL)) for further searching of words in the document file. Binary search algorithm is recommended for searching the word. Show the result, in a format as the above example with the average rank, on the screen as well as save them in a text file named **Result.txt**.

At least, but not limited to, two functions need to be implemented as specified below:

findRank(string *dictionary, string word) : Search the **word** in the ***dictionary** and return the rank of the **word**. Return the rank 1001 if the **word** is not in the ***dictionary**.

saveResult(int *rank, string outFileName) : The **int *rank** stores the rank of all words in the document file. The function saves the result to an output file **outFileName**.

■ *AUTOLAB Submission Check:*

```
double answer1;     // Store the average rank of Document.txt file
int answer2;        // Store the number of words in Document.txt which cannot be found in
                       the dictionary file Dictionary_1000.txt
```

Challenge Program: Pangram Checker (Bonus Points 25%)

A pangram or holoalphabetic sentence is a sentence using every letter of a given alphabet at least once. Pangrams have been used to display typefaces, test equipment, and develop skills in handwriting, calligraphy, and keyboarding. The best known English pangram is "The quick brown fox jumps over the lazy dog." It has been used since at least the late 19th century, was utilized to test Telex data communication equipment for accuracy and reliability, and is now used by a number of computer programs to display computer fonts.

Design a function **PangramCheck** to determine whether an input string sentence is a pangram, and

write a main program to call the **PangramCheck** function to check the sentences in the file **PangramList.txt**. Find out the number of sentences which is not a pangram in the file and show these sentences on the screen.

■ *AUTOLAB Submission Check:*

```
int answer1;    // Store the total number of sentences which is NOT a pangram in the file.
int answer2;    // Store the total number of sentences which is a pangram in the file.
```

Notes:

1. Please submit your programs (source codes) to the AUTOLAB grading system website (http://140.112.183.225) before **Dec. 24** (3:30PM)

2. Late submission will have a penalty of 10% discount per day of your homework total score toward a maximum of 50% discount. No late submission over five days will be accepted.

3. Criteria of grading include: (1) Program functionality; (2) User interface; (3) Structure of the program; (4) Suitable comments; (5) Programming style; (6) Creativity.