Computer Programming Language

【Fall, 2020】

Homework 7

Program A： Streams and File I/O (50%)

The text file **words.txt**, which is provided on our CEIBA course website, contains an alphabetically sorted list of English words. Note that the words are in mixed upper and lowercase.

Write a program that reads this file and finds the longest word that reverses to a different word. For example, "stun" reverses to make the word "nuts" but is only four letters long. Find the longest such word and show it on the screen.

You need to design your program with a modular approach using functions.

■ *AUTOLAB Submission Check:*

int answer1; // Store the total number of words in the **words.txt** file.

int answer2; // Store the number of characters of the longest word you find.

Program B： Classes – Movie Rating Survey Analysis (50%)

In this program you need to use Object-Oriented Programming approach to design a program for movie rating survey analysis. Your program need to read in several hundreds of movie ratings from a survey input file **RatingList.txt**, and output the analysis containing a list of movie title, number of ratings, and average rating of each movie to a file **RatingResult.txt**.

You should not assume you know how many rows (lines) are in the input file. Each line in the **RatingList.txt** represents one review, containing the movie title and its rating separated by a '|' character. However, you can assume that there are at most 20 movie titles in the input file.

To analyze the average rating of each movie using the OOP approach, you may define a class *MovieReview* that contains 4 private data members to hold the four data *string movieTitle, int totalScore, int numRating*, and *int aveScore*. The *MovieReview* class should have public accessor and mutator functions, *getMovieTitle*(), *getTotalScore*(), *getNumRating*(), *getAveScore*(), *setMovieTitle*(), *setTotalScore*(), *setNumRating*(), and *setAveScore*(). The MovieReview class should also have public member functions *addScore*() and *calculateAveScore*() to calculate the average rating of each movie.

The main program should have a function called *readData*() that reads in the data from the input file into an static array of *MovieReview* class objects that contain the movie title and the rating information, assuming that there are at most 20 movie titles in the input file. However, since you don't know how many rows are there in the file, you need to apply dynamic memory allocation method, using the *new* and *delete* operators, to store the rating data for later processing.

■ *AUTOLAB Submission Check:*

```
int answer1;      // Store the total number of movies listed in the RatingList.txt file.
int answer2;      // Store the total number of ratings in the RatingList.txt file.
string answer3;   // Store the movie title with the highest average rating.
```

Challenge Program C (Optional): K-Means Clustering Algorithm (Bonus Points 30%)

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups (clusters) in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are: the centroids of the K clusters (which can be used to label new data), and labels for the training data (each data point is assigned to a single cluster).

The pseudocode of the K-means clustering is summarized as follows:

```
1.  Clusters the data into K groups where K is predefined.
2.  Select K points at random as cluster centers.
3.  Assign objects to their closest cluster center according to the Euclidean distance function.
4.  Calculate the centroid or mean of all objects in each cluster.
5.  Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.
```

Write a program that reads the **Iris.data** text data file which contains the petal length and petal width of three speices of Iris flower in the first and second column. Let K = 3, use the K-means algorithm to label the data and saved them into a new text file named **Iris.out** with three labels A, B, and C, in the first column, respectively. Output the labeled data and the 3 cluster centroids on the screen.

■ *AUTOLAB Submission Check:*

```
int answer1;   // Store the number of data in cluster A.
int answer2;   // Store the number of data in cluster B.
int answer3;   // Store the number of data in cluster C.
```

Challenge Program D (Optional): Sudoku (Bonus Points 30%)

Solve a partially filled-in normal 9×9 Sudoku grid and display the result in a human-readable format. Backtracking algorithm, but not the only algorithm, is usually used to solve the Sudoku that iterates all the possible solutions for the given sudoku. If the solutions assigned do not lead to the solution of Sudoku, the algorithm discards the solutions and rollbacks to the original solutions and retries again and hence the name backtracking.

Below is the general pseudocode of backtracking algorithm for standard sudoku template (9×9):

```
Initialize 2D array with 81 empty grids (Row=9, Col= 9)
Fill in some empty grid with the known values
Make an original copy of the array
Start from top left grid (nx = 0, ny = 0), check if grid is empty
if (grid is empty) {
    assign the empty grid with values (i)
    if (no numbers exists in same rows & same columns same as (i) & 3x3 square (i) is currently in)
        fill in the number
    if (numbers exists in same rows & same columns same as (i) & 3x3 square (i) is currently in)
        discard (i) and repick other values (i++)
}
else {
    while (nx < 9) {
        Proceed to next row grid(nx++, ny)
        if (nx equals 9) {
            reset nx = 1
            proceed to next column grid(nx,ny++)
            if (ny equals 9) {
                print solution
            }
        }
    }
}
```

Your program should allow the user to input a Sodoku game to be solved via a text file **sudoku.txt** and then save the solution in an output file as well as displaying it on the screen. You also need to record and show the computation time of the solution process.

Test your program with the following puzzles representing three different difficulty levels. Compare the computation time of these three puzzles.

2			8		4			6
		6				5		
	7	4				9	2	
3				4				7
			3		5			
4				6				9
	1	9				7	4	
		8				2		
5			6		8			1

Puzzle A

		2						
		3		1				6
	4			2			3	
1					3			9
		5				4		
2			6					8
	9			7			4	
7				8		5		
						3		

Puzzle B

		2				4	3	
		3		6	5			
					8			
9						2		
	8						5	
		7						3
			4					
			9	8		3		
	5	6				7		

Puzzle C

■ *AUTOLAB Submission Check:*

```
int answer1;    // Store the number of center cell (5,5) of solved puzzle A.
int answer2;    // Store the number of center cell (5,5) of solved puzzle B.
int answer3;    // Store the number of center cell (5,5) of solved puzzle C.
```

Notes:

1. Please submit your programs (source codes) to the AUTOLAB grading system website (http://140.112.183.225) before **Jan. 7** (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.