CSC 230 – Project #2

Due: Tuesday, Mar. 9th, before midnight

Project Details:

It is tax season again! Let's design a software for the friendly IRS. This software stores all the data in **array**. Each person has **SSN** and **name** information. These information is provided in separated files. The software provides following basic operations for the data in the array:

- **Insertion**. The software adds the data to the **end** of the data in the array. If the provided data has duplicated SSN with an existing entry, the software will discard the provided data. Otherwise, increase the insertion counter by one.
- **Delete**. The software deletes the entry with given SSN and name from the array. It is possible that provided SSN does not match any record in the array, if that is the case, the software ignore this deletion request. If there is a match in the array, delete the record from the array, increase the delete counter by one.
- **Retrieval**. The software searches the given SSN and name in the array. If the there is a match in the array, increase the retrieval counter by one.

An input file looks like the following:

```
i 586412373        NICOLA EVANGELISTA
i 177228167            MEAGAN LEKBERG
i 586412373               JEFF DUTTER
i 760846483            KITTY MANZANERO
i 061899135        CATHERIN MCCREIGHT
i 087300880            CARMA KULHANEK
i 177264549         VALERY KOSAKOWSKI
i 210044984            SHEILAH MONGES
d 760846483            KITTY MANZANERO
r 760846483            KITTY MANZANERO
r 007980295            DELPHIA SIMISON
i 493515916            VERONIKA TADENA
d 401991909       MCKINLEY WESTERFELD
i 793267575             TEMIKA MESHEW
i 319373939              MARGIT EBLIN
```

In above file, each row represents one person's information. The leading character is either "i", "d", or "r". The second column is the SSN of the person, the following string are the first name and last name.

- The software should process the information row-by-row, from the beginning to the end of the file.
- The character "i" means insertion. The software should insert the SSN and name to the end of the data in the array. However, the software will NOT insert personal information with duplicate SSN. For example, the first row and the third row have duplicate SSN. The information of the first person will be added, but the third will not.
- The character "d" means deletion. In this example, there are two "d" rows. The first one has SSN `760846483,` which matches the fourth row. The corresponding record in the array will be deleted, the following records will be moved up by one position (just like part 2 of Lab3). The second "d" row does not match any existing SSN, the information of this row will be ignored.
- The character "r" means retrieval. If there is a match with the given SSN, increase the retrieval counter by one. In the above example, there are two "r" rows. The first one has a match, the second one does not have match.

In this project, we store the data in the array. Here are several requirements for the array:

- The initial size of the array is 1000.
- When you **add** a new entry to the array, if the array is not full, add the entry. If the array is full, creates a new array with **doubled size**. Copy the data from the old array to the new array. **Release** the old array.
- When you **delete** an entry from the array. If the number of entry is less than ¼ of the array size. Creates a new array with half size of the current array. Copy the data from the old array to the new array, then **release** the old array.

If the above example is stored in file 15-idr, when we run the program with the following command:

```
jli$ ./a.out 15-idr
The Number of Valid Insertation :10
The Number of Valid Deletion :1
The Number of Valid Retrieval :0
Item numbers in the array :9
Array Size is :500
Time elaspsed :0.000332
```

Almost all the output is self-explaining, except the last line, which tells how much time to execute the program. In your implementation, at the beginning of the main function, the program records the current time; at the end of the main function, the program records the current time again. The difference of these two values is the elapsed time. The timer.cpp file on Canvas shows how timer works. Please note different computer has different computing

capabilities, it is likely you will get different elapsed time value, even the input files are the same.

Several sample results are listed as follows:

```
jli$ ./a.out 20-id
The Number of Valid Insertation :17
The Number of Valid Deletion :2
The Number of Valid Retrieval :0
Item numbers in the array :15
Array Size is :250
Time elaspsed :0.000331
```

```
jli$ ./a.out 500-idr
The Number of Valid Insertation :327
The Number of Valid Deletion :27
The Number of Valid Retrieval :16
Item numbers in the array :300
Array Size is :500
Time elaspsed :0.005403
```

```
jli$ ./a.out 1000-idr
The Number of Valid Insertation :642
The Number of Valid Deletion :48
The Number of Valid Retrieval :42
Item numbers in the array :594
Array Size is :1000
Time elaspsed :0.014956
```

```
jli$ ./a.out 2000-idr
The Number of Valid Insertation :1278
The Number of Valid Deletion :100
The Number of Valid Retrieval :107
Item numbers in the array :1178
Array Size is :2000
Time elaspsed :0.058108
```

```
jli$ ./a.out 3000-idr
The Number of Valid Insertation :2202
The Number of Valid Deletion :74
The Number of Valid Retrieval :60
Item numbers in the array :2128
Array Size is :4000
Time elaspsed :0.145211
```

```
jli$ ./a.out 10000-idr
The Number of Valid Insertation :7190
The Number of Valid Deletion :259
The Number of Valid Retrieval :247
Item numbers in the array :6931
Array Size is :7936
Time elaspsed :1.47615
```

```
jli$ ./a.out 20000-idr
```

```
The Number of Valid Insertation :14453
The Number of Valid Deletion :481
The Number of Valid Retrieval :476
Item numbers in the array :13972
Array Size is :16000
Time elaspsed :6.03521
```

```
jli$ ./a.out 30000-idr
The Number of Valid Insertation :21725
The Number of Valid Deletion :759
The Number of Valid Retrieval :736
Item numbers in the array :20966
Array Size is :32000
Time elaspsed :13.5294
```

Questions to think about?

1. Why the elapsed time is not linearly increase? In other words, it is not proportional to the problem size (in this project, it is the entry number in the file)?
2. Which part of your program is the most time consuming? Why?

How many functions do I need? It is up to you. It depends on how you will divide functions among methods. But that does not mean you can/should just define one function, that is main(). If you do so, believe me, it is hard to do debugging. I prefer you define one function that checks whether the given SSN matches any entry in the array, one function does the insertion job, one function does the deleting job.

Do NOT make one method do too many things!!!!!

Anything else I should know about the method that checks match? Think about as a human being how to solve it, in a simplified situation. Then think about how to solve a more general case. Repeat this process until you have a solution for the original problem. Write down your solution (algorithm) on paper.

NEVER ever write code before you have a scrap paper with your solution on it. NEVER try to figure out your idea after you start typing. It will save you a lot of time.

Please build your code incrementally. Start coding with almost nothing inside, then add several lines, compile it, check it. If the code looks fine, add several more lines, compile it, check it. Repeat this process until the job is done.

When you decide which part should be implemented first, consider the simplest function of the project. Implement the simple ones first, handle the more complicated ones later. As human being, we prefer to solve simple questions first. The same rule applies to programming.

Never ever write down the whole code, then do the debugging. Remember we build a building piece-by-piece, floor-by-floor. No one builds all the floors of a building at the same time. It won't work!!!