Julius Ranoa
CSC 121 001 Computer Science I
Homework – Chapter 9 Searching, Sorting, and Algorithm Analysis

Part I. Review Questions @ Page 639 – 640.
Qn. 1 – 6, 11 – 18.

1. The *linear* search algorithm steps sequentially through an array, comparing each item with the search value.
2. The *binary* search algorithm repeatedly divides the portion of an array being searched in half.
3. The *linear* search algorithm is adequate for small arrays but not for large arrays.
4. The *binary* search algorithm requires that the array's contents be sorted.
5. The *average* number of comparisons performed by linear search to find an item in an array of N elements is *N / 2*.
6. The *maximum* number of comparisons performed by linear search to find an item in an array of N elements is *N*.
11. If an array is sorted in *ascending* order, the values are stored from lowest to highest.
12. If an array is sorted in *descending* order, the values are stored from highest to lowest.
13. Bubble sort places *one* number(s) in place on each pass through the data.
14. Selection sort places *one* number(s) in place on each pass through the data.
15. To sort N numbers, bubble sort continues making passes through the array until *no passes are made*.
16. To sort N numbers, selection sort makes *N – 1* passes through the data.
17. Why is selection sort more efficient than bubble sort on large arrays?
    Selection sort does fewer data swaps than the bubble sort.
18. Which sort, bubble sort or selection sort, would require fewer passes to sort a set of data that is already in the desired order? *Bubble Sort.*

Part II. Programming Challenge @ Page 644.
Qn. 15 – Using Files: String Selection Sort Modification.

Screenshot of Runtime:

```
Here is the sorted list.
  Allison, Jeff
  Collins, Bill
  Conroy, Pat
  Griffin, Jim
  Harrison, Rose
  Holland, Beth
  Johnson, Jill
  Kelly, Sean
  Michalski, Joe
  Moreno, Juan
  Moretti, Bella
  Patel, Renee
  Rubin, Sarah
  Sanchez, Manny
  Smith, Bart
  Smith, Cathy
  Taylor, Tyrone
  Whitman, Jean
  Wolfe, Bill
  Wu, Hong
End of list.

Process finished with exit code 0
```

Source Code:

1. *NameList.h*
2. *NameList.cpp*
3. *main.cpp*

The source code is also stored at Github.
Link below:

https://github.com/TheLoneWoof1102/FA17_CSC121001/tree/master/Source%20Code/Homework-Ch9.Qn15

main.cpp

```cpp
#include "NameList.h"
#include <iostream>
using namespace std;

int main() {
    NameList nl;

    if ( !nl.loadNames("names.dat") ) {
        cout << "Can't open file. :(" << endl;
        return -1;
    }

    nl.sortNames();
    cout << "Here is the sorted list." << endl;
    std::cout << nl.stringifyNames("  ");
    cout << "End of list." << endl;

    return 0;
}
```

// END of main.cpp.

NameList.h

```cpp
#ifndef HOMEWORK_CH9_QN15_NAMELIST_H
#define HOMEWORK_CH9_QN15_NAMELIST_H

#include <string>
#include <vector>

class NameList {

private:
    std::vector<std::string> names;
    bool isSorted = false;

public:
    NameList() {
        names.clear();
        isSorted = false;
    }
    bool sorted() {
        return isSorted;
    }
    void addName(std::string);
    bool loadNames(std::string);
    void sortNames();
    std::string stringifyNames(std::string = "");

};

#endif //HOMEWORK_CH9_QN15_NAMELIST_H
```

// END of NameList.h.

NameList.cpp

```cpp
#include "NameList.h"
#include <fstream>
#include <sstream>
using namespace std;

void NameList::addName(string name) {
    names.push_back(name);
}

bool NameList::loadNames(string filename) {
    ifstream data;
    string tempName;

    data.open(filename);
    if (!data) return false;

    while(data.good()) {
        getline(data, tempName);
        if (tempName.length() <= 0) continue;
        // To deal with return chars in Mac.
        if(tempName.back() == '\r') tempName.pop_back();
        addName(tempName);
    }

    return true;
}

void NameList::sortNames() {
    string first_val;
    int first_idx;

    for (int i = 0; i < names.size(); i++) {
        first_idx = i;
        first_val = names[i];
        for (int j = i + 1; j < names.size(); j++) {
            if (names[j] < first_val) {
                first_idx = j;
                first_val = names[j];
            }
        }
        if (first_val != names[i]) {
            names[first_idx] = names[i];
            names[i] = first_val;
        }
    }
}

string NameList::stringifyNames(string prefix) {
    stringstream ss;
    for (string name : names) {
        ss << prefix << name << endl;
    }
    return ss.str();
}
```

// END of NameList.cpp