First Name Last Name Section #

CSCE 221: Programming Assignment 2 (100 points)

Due February 20 by 11:59 pm

Project Cover Page

This project is a group project. For each group member, please print first and last name and e-mail address.

1.

2.

3.

Please write how each member of the group participated in the project.

1.

2.

3.

Please list all sources: web pages, people, books or any printed material, which you used to prepare a report and implementation of algorithms for the project.

Type of sources:	
People	
Web Material (give URL)	
Printed Material	
Other Sources	

I certify that I have listed all the sources that I used to develop solutions to the submitted project and code.

Your signature Typed Name Date

I certify that I have listed all the sources that I used to develop a solution to the submitted project and code.

Your signature Typed Name Date

I certify that I have listed all the sources that I used to develop solution to the submitted project and code.

Your signature Typed Name Date

Aggie Code of Honor: "An Aggie does not lie, cheat, or steal, or tolerate those who do."

You will be required to manage this project with private GitHub. For more information on this, see the Git tutorial.

STORY

Consider a situation that you have booked a flight from Houston to London. But due to unfortunate reasons, you reach the airport late. Now in this situation you look up to the board with flight details to check the status of your flight. If the flight board is not ordered in some way and random in nature it will look like this.

Flight Number	Destination	Departure Time	Gate Number
AA223	LAS VEGAS	21:15	A3
BA023	DALLAS	21:00	A3
AA220	LONDON	20:30	B4
VI303	MEXICO	19:00	A7
BA087	LONDON	17:45	A7
AA342	PARIS	16:00	A7
VI309	PRAGUE	13:20	F2
QU607	TORONTO	08:30	F2
AA224	SYDNEY	08:20	A7
AF342	WASHINGTON	07:45	A3

The above table is very tough to read. But say you can see another table which is sorted by Departure Time like this one:

Flight Number	Destination	Departure Time	Gate Number
AF342	WASHINGTON	07:45	A3
AA224	SYDNEY	08:20	A7
QU607	TORONTO	08:30	F2
VI309	PRAGUE	13:20	F2
AA342	PARIS	16:00	A7
BA087	LONDON	17:45	A7
VI303	MEXICO	19:00	A7
AA220	LONDON	20:30	B4
BA023	DALLAS	21:00	A3
AA223	LAS VEGAS	21:15	A3

This project will give you a better understanding of a sorting process.

PROJECT TECHNICALITIES

- Write a C++ program which reads the table of flights into a vector of `Flight` objects (of class `Flight`), and displays the sorted table based on the selected option: the departure time or the destination.

Steps:

1. Read the file with the flight details. Store each flight into a `Flight` class object. Create the vector of `Flight` objects and keep class members as strings, i.e., `flightNum`, `destination`, `departureTime` and `gateNum`.

2. Sort the flights using the selection, insertion, and bubble sort algorithms.

   In the sorted table, the selected sorting option (the departure time or the destination) should be in lexicographical (ascending) order.

3. Display the final sorted data in a table format.

   Remember that positions in the non-sorted columns will also change based on the sorted column. In each iteration of the sorting algorithm, elements change their position in the vector. For example in the selection sort, after the i-th iteration, the (i-1)-st element of the vector is swapped with the smallest element among i-th to n-th elements. The format of the output file is the same as the format of the input file.

Program Implementation Requirements

- Use the provided `flight.h` file to complete the `flight.cpp` file with the required fields.
  - Complete the `compareToDestination()` and `compareToDepartureTime()` functions to compare two `Flight` objects based on the departure time or destination.
- Implement the `selection_sort`, `bubble_sort`, and `insertion_sort` functions defined in the supplemental file `sort.h`.
  - These will take an arbitrary vector of `Flight` objects and sort them. These functions will also take in an argument to switch between sorting by destination or by departure time.
- Divide the problem in three steps: reading from the given input file, sorting the table, and writing the sorted table to a file.
  - You can find information about input and output streams in "*Programming. Principles and Practice Using C++.*" by B. Stroustrup, 2nd edition, Chapters 10.4-11 and 11.4-5.
  - You should use the `getline` function (from `std::string` class) with two arguments to read lines of the `csv` files:

    ```
    istream& getline (istream& is, string& str);
    ```

  - Next, use `getline` function with three arguments, where the third argument is a delimiter character, to parse a line, see this description:

    ```
    istream& getline (istream& is, string& str, char delim);
    ```

which extracts characters from `is` and stores them into `str` until the delimitation character `delim` is found. Use comma (`,`) as a delimiter. To split strings into tokens you can use the C function `strtok()`, or preferably regular expressions, see "*Programming. Principles and Practice Using C++*" by B. Stroustrup, 2nd edition, Chapters 23.6–23.9.

- Insert the code that increases number of comparison `num_cmps++` typically in an `if` or a loop statement
  - This variable will be defined in `sort.h`, but you have to provide the code to manage this variable.
  - Remember that C++ uses the shortcut rule for evaluating boolean expressions. A way to count comparisons accurately is to use comma expressions. For instance,

    ```
    while (i < n && (num_cmps++, a[i] < b)) ...
    ```

File Format Requirements

- The file will have the same format as given, but will have different data.

  Here is a sample:

  ```
  FLIGHT NUMBER,DESTINATION,DEPARTURE TIME,GATE NUMBER
  BC580,McMechen,12:33,Z3
  ZK612,Glenrock,05:27,Y8
  BF144,South Range,01:33,R4
  US141,Tate,17:06,J5
  JO745,Edmundson,21:07,V4
  ```

- Fist line will consist of the column headers.
- Each subsequent line will have a flight details.
- The cvs files will contain details for sets of 10, 100, 1000, and 10000 flights. These files are split into sets with ascending, descending, and random order.
- Flight information format:
  - Flight Number: 2 alphabetic characters followed by a three digit integer.
  - Destination: Name of a city (can be many words)
  - Departure: 24 hour format.
  - Gate Number: 1 alphabetic character followed by a single digit integer.
- Check the input file format for correctness and throw an exceptions if the format is not preserved.
- Write the sorted table to a file and display on a terminal for smaller inputs (10 flights only).

Submission Requirements for Source Code and Report

Turn in an electronic version of your C++ code, your report and testing file to eCampus as a tar file.

1. **C++ code requirements:**

   (a) At the top of your source file containing the function `main()` (the first few lines) write a comment containing personal information of a group members, including course number, course section, your first and last name, and email address. If you do not provide this information, points will be deducted. This is the format which is required:

```
/************************************
Programming Assignment 2 CSCE 221-5xx
Your Last Name(s), First Name(s)
your email(s)
************************************/
```

   (b) Use consistent indentation. Your program should be readable – select variable names wisely and document your code. Get more information about the programming style, read http://www.stroustrup.com/Programming/PPP-style-rev3.pdf

2. **The report requirements.** Return an electronic and hard copy of the report followed by the cover page including answers to provided questions.

   (a) A brief description of assignment purpose, its description, the input and output files format, instructions how to run your program.

   (b) Program design and the class definitions. Explanation of splitting the program into classes and *a description of C++ object oriented features or generic programming used in this assignment*.

   (c) C++ object oriented features like input/output operators overloading or constant member functions.

   (d) **Algorithms.** Briefly describe the features of each of the sorting algorithms.

   (e) **Theoretical Analysis.** Provide the theoretical number of comparisons perform on elements of the vector and big-O notation of the sorting algorithms on input of size n in decreasing (dec), random (ran) and increasing (inc) orders and fill the tables below.

# of comps.	best	average	worst
Selection Sort			
Insertion Sort			
Bubble Sort			

big-O notation	inc	ran	dec
Selection Sort			
Insertion Sort			
Bubble Sort			

   (f) **Experiments and Statistics.** Briefly describe the experiments. Present the number of comparisons (**#COMP**) performed on input data using the following tables.

#COMP	Selection Sort			Bubble Sort			Insertion Sort		
n	inc	ran	dec	inc	ran	dec	inc	ran	dec
10									
10^2									
10^3									
10^4									

   inc: increasing order; dec: decreasing order; ran: random order

   (g) For each sorting algorithm, graph the number of comparisons versus the input size $n = 10^4$, totaling in 9 graphs over the three input cases (inc, ran, dec) versus the input size for the three algorithms.

   HINT: To get a better view of the plots, *use logarithmic scales* for both x and y axes.

(h) Testing cases for $n = 10$, supported by screen shots (remember to use white background to save ink)

(i) **Discussion.** Comment on how the experimental results relate to the theoretical analysis and explain any discrepancies you note. Do your computational results match the theoretical analysis you learned from class or the textbook? Justify your answer.

(j) **Conclusions.** Give your observations and conclusion. For instance, which sorting algorithm is more suitable for a specific input data? What factors can affect your experimental results?

Turning In

1. Use `tar` program to bundle all your files in one file.
2. Do not turn in the object files and/or executable file.
3. Note: Late projects are penalized according to the weights provided in the syllabus.
4. If your program does not compile on a Linux machine or if there are run-time errors, you will get at most 50% of the total number of points.