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. Hunter Cleary hncleary@tamu.edu 625001547

2. Jeremy Brown jjbrown17@hncleary.edu 826008277

3.

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

1. Hunter Cleary - Analysis of experiment / graphing and tables, sorting algorithms creation

2. Jeremy Brown - Flight class implementation, sorting algorithms creation

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)	https://en.wikipedia.org/wiki/Insertion_sort https://en.wikipedia.org/wiki/Bubble_sort https://en.wikipedia.org/wiki/Selection_sort
Printed Material	Data Structures and Algorithms (Textbook)
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 Hunter Cleary 2-17-18

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

Your signature Jeremy Brown 2-17-18

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

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

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

   Answer
   The purpose of this assignment is to properly sort flight data using multiple sorting algorithms on two different parameters. Three sorting algorithms will be utilized: selection sort, insertions, and bubble sort. Code will be tested for a variety of inputs. Running time and and comparison counts will be analyzed to see correlation between results and the variation of the input (size / relative order). Input and output files will be in CSV format.

   After the code submitted is compiled using the makefile, the rand10.csv is inputted and then sorted 6 times using the 3 algorithms and the 2 enumerators. The output is a CSV file with 6 sets of sorted values. Each set is labeled to indicate which algorithm / enumerator was used.

2. 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*.

   Answer
   The program is split between flight.cpp, sort.cpp, and PA2.cpp. The flight class contains a structure to store string values and declares functions used to compare destination and departure time strings. Sort.cpp, sort.h receive a vector of flights and an enumerator. Utilizing the enumerator for each, the flights are sorted using one of the three sort functions. Object oriented programming allows for the use of enumerators to decide what parameters to sort by.

3. C++ object oriented features like input/output operators overloading or constant member functions.
   Answer

4. **Algorithms.** Briefly describe the features of each of the sorting algorithms.

   Selection Sort
   Selection sort is a basic algorithm that is inefficient on large sets of data. The algorithm divides data into the sorted and unsorted. Comparisons are made to find the smallest / largest element of the unsorted values, then that element is placed in the proper place of the list. This algorithm will always make the same number of comparison regardless of the initial order of the elements.

   Insertion Sort
   Insertion sort is an algorithm that builds the final sorted list one item at a time. The algorithm takes each subsequent item in a set and then compares it to each value that has been

sorted to find where it belongs. It is most useful when the items in a set are already partially sorted. The worst case for this algorithm occurs when all elements are in reverse order.

Bubble Sort

The bubble sort algorithm repeatedly steps through a set a data, making comparisons on points adjacent to one another. These points are then reordered based on the desired output. It is most effective when the data set only has a few out of place elements that are close to their proper location. Akin to insertion sort, the worst case scenario occurs when the elements are in reverse order.

5. **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	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$

big-O notation	inc	ran	dec
Selection Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$

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

Using Departure Time Enumerator

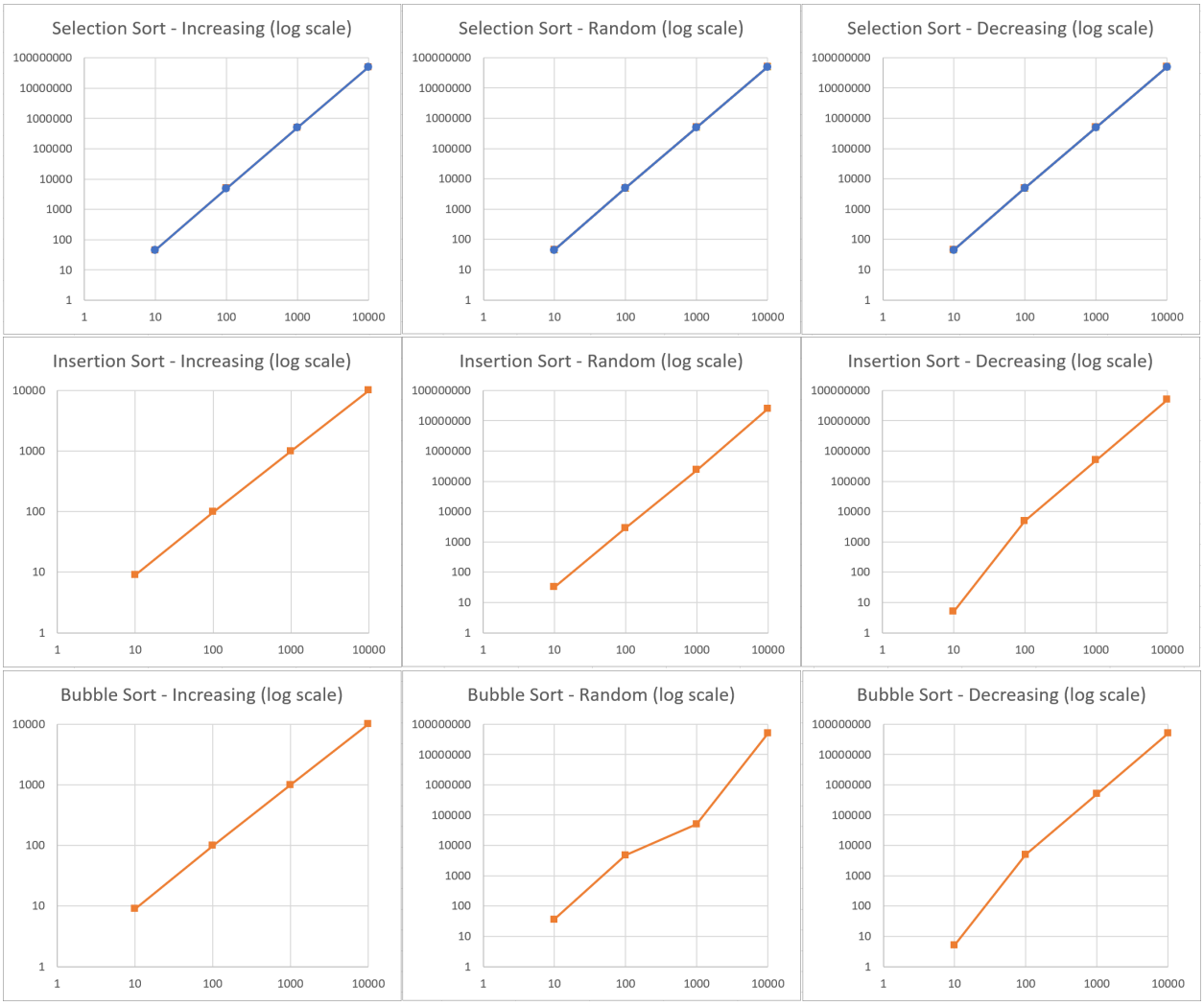
#COMP	Selection Sort			Bubble Sort		
n	inc	ran	dec	inc	ran	dec
10	45	45	45	9	36	45
10^2	4950	4950	4950	99	4796	4950
10^3	499500	499500	499500	999	49817	499500
10^4	50004999	50004999	50004999	9999	49994994	49995000

#COMP	Insertion Sort		
n	inc	ran	dec
10	9	32	45
10^2	99	2911	4950
10^3	999	238241	499500
10^4	9999	25098813	49995000

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

7. 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.


Selection Sort - Increasing (log scale)
Selection Sort - Random (log scale)
Selection Sort - Decreasing (log scale)
Insertion Sort - Increasing (log scale)
Insertion Sort - Random (log scale)
Insertion Sort - Decreasing (log scale)
Bubble Sort - Increasing (log scale)
Bubble Sort - Random (log scale)
Bubble Sort - Decreasing (log scale)

8. Testing

```
Flight Number,Destination,Departure Time,GateNumber
GB730,Abbeville,00:00,V1
GL220,Abbotsford,02:23,D3
CJ227,Abbott,04:47,U2
AB036,Abbottsburg,07:11,A1
AG739,Abbottstown,09:35,W1
HD243,Abbyville,11:59,R0
SR755,Abell,14:23,C1
UP401,Abercrombie,16:47,E2
XT369,Aberdeen,19:11,P8
BC298,Aberfoil,21:35,Z4
```

ByDeparture Increasing

```
Flight Number,Destination,Departure Time,GateNumber
GB730,Abbeville,00:00,V1
GL220,Abbotsford,02:23,D3
CJ227,Abbott,04:47,U2
AB036,Abbottsburg,07:11,A1
AG739,Abbottstown,09:35,W1
HD243,Abbyville,11:59,R0
SR755,Abell,14:23,C1
UP401,Abercrombie,16:47,E2
XT369,Aberdeen,19:11,P8
BC298,Aberfoil,21:35,Z4
```

ByDestination Increasing

```
Flight Number,Destination,Departure Time,GateNumber
HL875,Likely,05:33,T9
EA731,Popejoy,06:25,C6
LQ237,Caney,09:00,G3
WL004,Hesler,10:16,L7
JD033,Tacoma,15:31,A5
BZ911,Warrensburg,15:38,Z0
GY885,Wind Lake,16:54,T2
MT099,Michie,19:17,Q2
JF326,Batavia,19:53,W6
OY170,Upson,21:37,E2
```

ByDeparture Random

```
Flight Number,Destination,Departure Time,GateNumber
JF326,Batavia,19:53,W6
LQ237,Caney,09:00,G3
WL004,Hesler,10:16,L7
HL875,Likely,05:33,T9
MT099,Michie,19:17,Q2
EA731,Popejoy,06:25,C6
JD033,Tacoma,15:31,A5
OY170,Upson,21:37,E2
BZ911,Warrensburg,15:38,Z0
GY885,Wind Lake,16:54,T2
```

ByDestination Random

```
Flight Number,Destination,Departure Time,GateNumber
PE602,Zita,02:23,P1
YL986,Zoar,04:47,G6
ZG694,Zolfo Springs,07:11,F4
WK628,Zona,09:35,H7
AB740,Zumbro Falls,11:59,V6
LR661,Zumbrota,14:23,P8
MD227,Zuni,16:47,N2
II025,Zurich,19:11,F2
BD531,Zwingle,21:35,A7
RN715,Zwolle,23:59,P2
```

ByDeparture Decreasing

```
Flight Number,Destination,Departure Time,GateNumber
PE602,Zita,02:23,P1
YL986,Zoar,04:47,G6
ZG694,Zolfo Springs,07:11,F4
WK628,Zona,09:35,H7
AB740,Zumbro Falls,11:59,V6
LR661,Zumbrota,14:23,P8
MD227,Zuni,16:47,N2
II025,Zurich,19:11,F2
BD531,Zwingle,21:35,A7
RN715,Zwolle,23:59,P2
```

ByDestination Decreasing

9. **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.

   The experimental results fell closely in line with the theoretical analysis. Best, average, and worst cases corresponded with increasing, random, and decreasing data sets. For selection sort all three cases remained the same, which correlated to the theoretical analysis that stated the best, average, and worst cases would always be equal $O(n^2)$. Bubble and insertion also ran as expected. Both of these algorithms perform best when used on data sets that are already partially sorted. In both best cases tested where the values were already sorted, the algorithm was only required to make comparisons until it was determined that the values were already ordered $O(n)$. In worst cases, every item had to be compared to every item and was equivalent to $O(n^2)$.

10. **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 result?

    Selection sort is a basic algorithm that would only be useful on very small inputs. Insertion sort and bubbles both have similar ideal situations. They work best on inputs that have already been somewhat sorted. A factor that could affect experimental results would be the order of the random values. Ideally, multiple sets of random values would need to be tested in order to find a proper average.

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.