CST238 Fall 2019

Lab 09 - Sorting Algorithms

Part 1. (10 points)

1. Create a Student class with a name and an id as member variables
2. Write getter functions for both attributes
3. Make sure to use an interface (Student.h) file with a header guard and a separate implementation (Student.cpp)
4. In main.cpp, write a readStudents function with the following signature:
   ```
   Student * readStudents(string filename, int & size);
   ```
5. Implement that function using the following format and example file; the first line is the number of students, and each subsequent line is a student
   ```
   5
   Borg 1123
   Wall-E 2002
   Robby 345
   Alita 4040
   Maximillian 2001
   ```
6. Write a printStudents function with the following signature and example output
   ```
   void printStudents(Student * sa, int size);

   Student roster:
       1. Borg 1123
       2. Wall-E   2002
       3. Robby    345
       4. Alita    4040
       5. Maximillian  2001
   ```
7. At the end of your main function, free the memory for the array of Student objects

Part 2. (20 points)

For this part, you will implement insertion sort. You can use class notes (lecture slides), but you cannot look up insertion sort code. You should only look at pseudocode or algorithm descriptions.

1. Add a new method to main to sort the array of Student objects *by ID, ascending*
   ```
   void sort(Student * sa, int size);
   ```
2. Implement the sort function using insertion sort
3. Call the sort function in main, after loading the array
4. Call the printStudents function after sorting

Once you are done, test each function, as with the main() function below:

```
int main() {
  string filename = "t1";
  int size = 0;
  Student * s = readStudents(filename, size);
```

```
  printStudents(s, size);
  sort(s, size);
  printStudents(s, size);
  return 0;
}
```

If everything is implemented correctly, you should not have any memory leaks, and should print:

```
Student roster:
    1. Borg 1123
    2. Wall-E   2002
    3. Robby    345
    4. Alita    4040
    5. Maximillian  2001

Student roster:
    1. Robby    345
    2. Borg 1123
    3. Maximillian  2001
    4. Wall-E   2002
    5. Alita    4040
```

Extra Credit (2 points): Before leaving the sort function, print out the number of swaps and comparisons (see example below, for data above). The numbers must be accurate for full credit.

```
Sorting Complete - Swaps: 4 Comparisons: 7
```