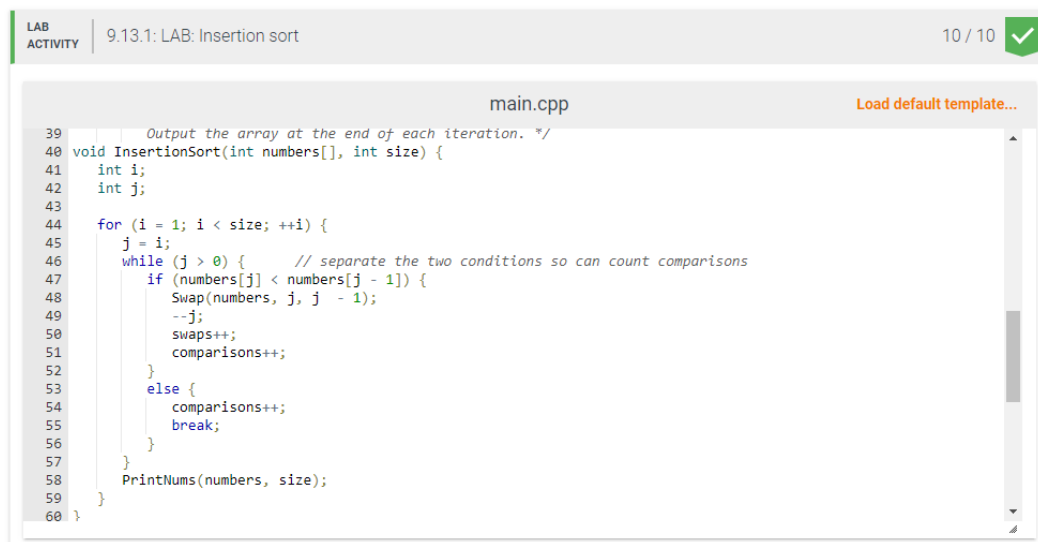


# CSCI 140 PA 9 Submission

Due Date: 5/2/23

Name: Ali Mortada

Exercise 1 – **zyBook 9.13 LAB: Insertion sort** -- check if completely done in zyBook  
\_\_X\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score

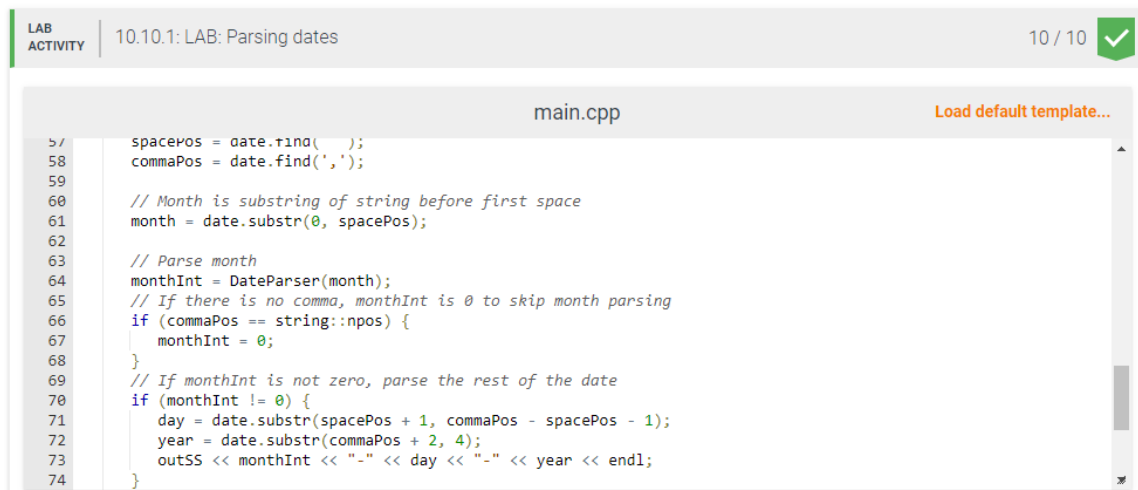


LAB ACTIVITY | 9.13.1: LAB: Insertion sort | 10 / 10 ✓

main.cpp Load default template...

```
39 // Output the array at the end of each iteration. */
40 void InsertionSort(int numbers[], int size) {
41     int i;
42     int j;
43
44     for (i = 1; i < size; ++i) {
45         j = i;
46         while (j > 0) { // separate the two conditions so can count comparisons
47             if (numbers[j] < numbers[j - 1]) {
48                 Swap(numbers, j, j - 1);
49                 --j;
50                 swaps++;
51                 comparisons++;
52             }
53             else {
54                 comparisons++;
55                 break;
56             }
57         }
58         PrintNums(numbers, size);
59     }
60 }
```

Exercise 2 – **zyBook 10.10 LAB: Parsing dates** -- check if completely done in zyBook  
\_\_X\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score



LAB ACTIVITY | 10.10.1: LAB: Parsing dates | 10 / 10 ✓

main.cpp Load default template...

```
57 // spacePos = date.find(' ');
58 // commaPos = date.find(',');
59
60 // Month is substring of string before first space
61 month = date.substr(0, spacePos);
62
63 // Parse month
64 monthInt = DateParser(month);
65 // If there is no comma, monthInt is 0 to skip month parsing
66 if (commaPos == string::npos) {
67     monthInt = 0;
68 }
69 // If monthInt is not zero, parse the rest of the date
70 if (monthInt != 0) {
71     day = date.substr(spacePos + 1, commaPos - spacePos - 1);
72     year = date.substr(commaPos + 2, 4);
73     outSS << monthInt << "-" << day << "-" << year << endl;
74 }
```

Exercise 3 – **zyBook 10.12 LAB\*: Program: Data visualization** -- check if completely done in zyBook \_\_\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score

Exercise 3 – **Timing Sorting Algorithm** -- check if completely done \_\_X\_\_ ; otherwise, discuss issues below

Source code below:

```
/*
    Program: Timing Sorting Algorithm
    Author: Ali Mortada
    Class: CSCI 140
    Date: 5/4/23
    Description: Times the selection sort of two text files containing 10k values and
    50k values. Shows that the runtime of selection sort is  $O(N^2)$ .
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

void selectionSort(int numbers[], int numbersSize);

int sortTime = 0;

int main() {

    ifstream inFS;
    ifstream inFS2;
    string value;
    string values10k[10000];
    string values50k[50000];
    int numValues = 0;

    cout << "Author: Ali Mortada" << endl << endl;

    // Attempt to open medium10k.txt
    cout << "Sorting file: medium10k.txt" << endl;
    inFS.open("medium10k.txt");
```

```

if (!inFS.is_open()) {
    cout << "Could not open file medium10k.txt." << endl;
    return 1;
}

// Count number of values and input values into array
for (int i = 0; i < 10000; i++) {
    getline(inFS, value);
    values10k[i] = value;
    numValues++;
}
cout << "Number of values: " << numValues << endl;

// Output first and last 5 unsorted values
cout << "First 5 (unsorted): ";
for (int i = 0; i < 5; i++) {
    cout << values10k[i] << " ";
}
cout << endl;

cout << "Last 5 (unsorted): ";
for (int i = 9995; i < 10000; i++) {
    cout << values10k[i] << " ";
}
cout << endl;

// Convert values10k to an int array
int intvalues10k[10000];
for (int i = 0; i < 10000; i++) {
    intvalues10k[i] = atoi(values10k[i].c_str());
}

// Sort medium10k.txt
selectionSort(intvalues10k, 10000);

// Output first and last 5 sorted values
cout << "First 5 (sorted): ";
for (int i = 0; i < 5; i++) {
    cout << intvalues10k[i] << " ";
}
cout << endl;

cout << "Last 5 (sorted): ";
for (int i = 9995; i < 10000; i++) {
    cout << intvalues10k[i] << " ";
}

```

```

}
cout << endl;

// Output sort time
cout << "Sort time: " << sortTime << " ms" << endl;

cout << endl << endl;

// Close medium10k.txt
inFS.close();

// Reset number of values and sort time
numValues = 0;
sortTime = 0;

////////////////////////////////////

// Repeat for medium50k
cout << "Sorting file: medium50k.txt" << endl;
inFS2.open("medium50k.txt");
if (!inFS2.is_open()) {
    cout << "Could not open file medium50k.txt." << endl;
    return 1;
}

for (int i = 0; i < 50000; i++) {
    getline(inFS2, value);
    values50k[i] = value;
    numValues++;
}

cout << "Number of values: " << numValues << endl;

cout << "First 5 (unsorted): ";
for (int i = 0; i < 5; i++) {
    cout << values50k[i] << " ";
}
cout << endl;

cout << "Last 5 (unsorted): ";
for (int i = 49995; i < 50000; i++) {
    cout << values50k[i] << " ";
}
cout << endl;

int intvalues50k[50000];

```

```

    for (int i = 0; i < 50000; i++) {
        intvalues50k[i] = atoi(values50k[i].c_str());
    }

    selectionSort(intvalues50k, 50000);

    cout << "First 5 (sorted): ";
    for (int i = 0; i < 5; i++) {
        cout << intvalues50k[i] << " ";
    }
    cout << endl;

    cout << "Last 5 (sorted): ";
    for (int i = 49995; i < 50000; i++) {
        cout << intvalues50k[i] << " ";
    }
    cout << endl;

    cout << "Sort time: " << sortTime << " ms" << endl;

    cout << endl << endl;

    inFS2.close();

    return 0;
}

void selectionSort(int numbers[], int numbersSize) {
    int i, j, indexSmallest, temp;

    for (i = 0; i < numbersSize; i++) {
        indexSmallest = i;
        for (j = i + 1; j < numbersSize; j++) {
            if (numbers[j] < numbers[indexSmallest]) {
                indexSmallest = j;
            }
            sortTime++;
        }

        temp = numbers[i];
        numbers[i] = numbers[indexSmallest];
        numbers[indexSmallest] = temp;
    }
}

```

Input/output below:

```
Author: Ali Mortada

Sorting file: medium10k.txt
Number of values: 10000
First 5 (unsorted): 54044 14108 79294 29649 25260
Last 5 (unsorted): 86901 29810 84151 9752 86952
First 5 (sorted): 5 14 56 57 63
Last 5 (sorted): 99939 99965 99971 99975 99988
Sort time: 49995000 ms

Sorting file: medium50k.txt
Number of values: 50000
First 5 (unsorted): 54044 14108 79294 29649 25260
Last 5 (unsorted): 84912 81480 38702 46740 72774
First 5 (sorted): 1 5 8 10 11
Last 5 (sorted): 99988 99989 99995 99998 100000
Sort time: 1249975000 ms
```

Medium50k is 5 times as large, so it takes around 25 times the amount of time of medium10k.

Answer for Question 1

Algorithm analysis allows the programmer to know how long it takes to run a certain algorithm. This can be helpful when determining which algorithm to use, especially when dealing with large data sets. Big O notation simplifies the runtime of a certain algorithm which helps in the vetting process. Thus, algorithm analysis is very helpful in optimizing a program to run as efficiently as possible.

Answer for Question 2

One good reason to use file input is for testing a program. Instead of having to input values into the console every time a program is run to test it, one can put all the input values into a text file and have the program read from the text file instead of the console. This makes it much easier to test a program, as all it takes to change an input is to change the content of the text file. Similarly, file output can be used to simplify program testing. Instead of writing to the console, a program can write to a text file, which could make analyzing the output easier. One program could write to a text file, and another program could read that text file to perform its own functions with it. Thus, file input and output have many uses that can make program testing much easier.

Extra Credit – provide if applicable.