CS 311 Data Structures and Algorithms, Fall 2020

Midterm Exam

Due 2 pm Friday, October 9, 2020

Name: ____

The exam is worth 120 points. 7 pages (including this cover page), 8 problems.

Instructions

Answer all of the problems on the following pages.

This exam is to be done individually. You may reference an class materials, books, your own notes, or anything on the web, but not other people.

Turn in a PDF on UAF Blackboard, under "Midterm Exam" for this class. The PDF should have your name on the first page. It should be turned in by 2 pm Friday, October 9, 2020.

1. [15 pts total] **Order of Useful Operations.** In each part, indicate the order of the given operation, first using Θ, and then, *based on this*, in words (e.g., "exponential time"). Assume that a good algorithm is used—within the constraints given. Use *n* to denote the number of items in the list that the algorithm is given as input. The model of computation is our usual one.

1a. [3 pts] Printing the last item in an array whose size is known.

Θ:

In Words:

1b. [3 pts] Printing the last item in a Linked List whose size is known.

Θ:

In Words:

1c. [3 pts] Finding an item with a given value in a sorted array.

Θ:

In Words:

1d. [3 pts] Sorting a Linked List.

Θ:

In Words:

1e. [3 pts] Sorting an array with **Insertion Sort**.

Θ:

In Words:

2. [5 pts] **Iterators & Ranges.** Vector `src` has size 8. It contains, in order, the values 0, 10, 20, 30, 40, 50, 60, 70. The following function call copies certain items in `src` to vector `dest`. List the **values** of the items of `src` that are copied.

```
std::copy(begin(src)+2, begin(src)+5, begin(dest));
```

3. [12 pts total] **Overloaded Operators.** For each of the following operators, indicate whether we should declare it as a member or a global function and how each of the parameter(s) should be passed (by value, etc.). *If you wish, you may answer each part by giving a prototype.*

3a. [4 pts] `operator+=`

3b. [4 pts] Binary `operator+`

3c. [4 pts] `operator<<` (stream output)

4. [8 pts total] **Ownership & Releasing.** Give 2 examples of resources that can be owned, along with what it means to *release* each one. An example is given.

Example resource: An open file
What it means to release the example resource: Close the file

Resource #1:

What it means to release resource #1:

Resource #2:

What it means to release resource #2:

5. [21 pts total] **Order of Functions.** Write the order of each function below, using big-*O*. The model of computation is our usual one.

5a. [7 pts]

```
void func_a(int arr[], int n) // n is size of arr
{
    cout << "Item 0 = " << arr[0] << endl;
    cout << "Item 1 = " << arr[1] << endl;
    cout << "Item 2 = " << arr[2] << endl;
}
```

Order (big-*O*):

5b. [7 pts]

```
void func_b(int arr[], int n) // n is size of arr
{
    for (int a = 0; a < n; ++a)
        for (int b = 0; b < a; ++b)
               arr[b] += arr[a];
}
```

Order (big-*O*):

5c. [7 pts] Function `func_c` takes a range specified with 2 random-access iterators. You can let *n* be the size of this range.

```
template<typename RAIter>
void func_c(RAIter first, RAIter last)
{
    for (RAIter it = first; it != last; ++it)
    {
        *it *= 2;
    }
}
```

Order (big-*O*):

6. [17 pts total] **The Master Theorem.** The following recursive function `gg` takes a range specified with 2 random-access iterators. The range must contain `int`s.

```
template<typename RAIter>
int gg(RAIter first, RAIter last)
{
    auto size = last - first;  // size of range (call this "n")

    // BASE CASE

    if (size <= 1)
        return 0;

    // RECURSIVE CASE

    // Add up values in range
    int total = 0;
    for (RAIter it = first; it != last; ++it)  // Some EXTRA WORK!
        total += *it;

    // Do a recursive call & return something
    auto middle = first + size/2;  // Iterator to middle of range
    if (total > 0)
        return total + gg(first, middle)
    else
        return total + gg(middle, last);
}
```

Apply the Master Theorem to find the order of function `gg`, using our usual model of computation. As a guide through this process, answer each of the following.

b = _______; a = _______.

$f(n)$ is $\Theta(n^d)$. d = _______.

Compare a and b^d (circle one comparison operator): a < = > b^d.

Which case of the Master Theorem? (circle one):   1   2   3

Conclusion. The order of function `gg` is (simplify!):   Θ(_____________).

7. [21 pts total] **Trade-Offs.** In each part, two methods of achieving some goal are listed. For each method, give one advantage it has **over the other**.

7a. [7 pts] Passing by reference, passing by reference-to-const.

Advantage of passing by reference:

Advantage of passing by reference-to-const:

7b. [7 pts] Binary Search, Sequential Search.

Advantage of Binary Search:

Advantage of Sequential Search:

7c. [7 pts] Introsort, Merge Sort.

Advantage of Introsort:

Advantage of Merge Sort:

8. [21 pts total] **Possible?** In each part, your friend Egbert makes a claim about a new algorithm. Indicate whether Egbert's claim is possible or impossible (circle one). If it is possible, explain **how to do it**. If impossible, explain **how you know it is impossible**. *Hint. One of these is possible, and two are impossible.*

8a. [7 pts] Egbert says, "I have a new algorithm that can find the largest number in any given unsorted array, in logarithmic time."

POSSIBLE IMPOSSIBLE

8b. [7 pts] Egbert says, "I have a new general-purpose comparison sort that can sort a list in linear time, assuming that each item's position is no more than 100 away from its position in the final sorted list."

POSSIBLE IMPOSSIBLE

8c. [7 pts] Egbert says, "I have a new algorithm that can sort any list in linear time, given only an appropriate comparison function."

POSSIBLE IMPOSSIBLE