Week 7 Exercises Part1

Sandra Batista

1.1–1.2

Exercise1: Max Comparator

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/max_comparator.cpp

1. Implement a template max function to use a type and comparator:

   template <typename T, typename comp> T mymax(const T& a, const T& b, comp test);
2. Write a comparator to compare the size of 2 vectors of ints and returns true if and only if the first parameter vector is the larger sized vector
3. Write a comparator to compares the sum of two vectors of ints and returns true if and only if the first parameter vector is the vector with the larger sum
4. Complete main to instantiate your comparators and test your max function

Exercise 2: HW1 revisited – Sorting Character Blocks

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/CharBlockSort.cpp

1. Write a comparator that will return true if and only if the first parameter CharacterBlock has a smaller physical address
2. Inside main() sort the CharacterBlocks using your comparator and verify that it worked correctly.

Exercise 3: LL Selection Sort

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/selectionsort.h

1. selectionsort.h contains the Item struct for a doubly linked list and the functions findmin and LLSelectionSort
2. Complete the implementation of findMin and LLSelectionSort in selectionsort.cpp.
3. The implementation of LLSelectionSort should be recursive. (It is good practice, but you can try an iterative implementation also.)
4. findMin should find the minimum item and remove it from the list and return a shared ptr to it. If you would prefer to only return the pointer and remove it in LLSelectionSort that is also acceptable.
5. The main in selectionsort.cpp can be used to test your LLSelectionSort. It has memory leaks that you can fix as an exercise later…

Exercise 4: Quantile

Code: https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/smallest.cpp

1. What is the runtime of the **quantile** function?
2. Find the recurrence relation for the runtime.
3. Use a recursion tree or unrolling to solve the recurrence.
4. Then apply an asymptotic bound on the runtime.