Midterm

- Don't Panic.
- Write your name on every page.
- The midterm contains five problems. You have 60 minutes to earn 100 points.
- The midterm contains 8 pages, including this one.
- The midterm is closed book. You may bring one double-sided handwritten "cheat sheet" of A4 paper to the quiz. (You may not bring any magnification equipment!) You may **not** use a calculator, your mobile phone, or any other electronic device.
- Write your solutions for each problem on separate pieces of paper.
- Read through the problems before starting. Do not spend too much time on any one problem. **Difficulty is not necessarily correlated with number of marks.**
- Show your work. Partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem #	Name	Possible Points	Achieved Points
1	Museum of Valuable Liquids	20	
2	Duplicate Removal	20	
3	Merging Binary Heaps	20	
4	Buggy Code: Exploring Planet Nine	20	
5	Linear Data Structures	20	
Total:		100	

Name: ______________________________ Student id.: ______________________________

Problem 1. The Museum of Valuable Liquids. [20 points]

You have been recruited to help plan a robbery of the Museum of Valuable Liquids. This legendary Museum contains samples of a wide variety of liquids, from water to mercury to cobra venom. You have been given a list of every item in the Museum; each item on the list names the item (e.g. "water"), along with the total weight and value of the sample (202.1kg, \$25.20). Technicalities: no two samples in the Museum have the same name; all weights are positive; all values are positive.

Because of the daring nature of the heist, it is not possible to carry away everything: you have a given maximum capacity C in kg. Of course, it's a bad idea to mix cobra venom and milk (among other things), so you'll need some flasks to carry the various liquids out of the Museum. Fortunately, you've acquired some flasks that are so light that you can ignore their weight. You do not need to take an entire sample: for example, rather than taking the entire "water" sample above, you could instead take half, for a weight of 101.05kg and a value of \$12.60. Of course, you can't take more of an item than the Museum actually has.

Your task: design an algorithm that will determine which samples (and how much of each) you should take to maximize the value your robber gang extracts from the Museum. Please

(a) [5 points] explain your algorithm in pseudocode;

(b) [10 points] give a big-O runtime of your algorithm, together with an explanation, where N is the number of items in the museum; and

(c) [5 points] explain why your algorithm will produce an optimal answer.

You can assume that the data comes to you in some convenient way, but should carefully explain any such assumptions. If you want to use an algorithm and/or data structure we have covered in class as part of your solution, then you don't have to explain how it works as long as you're using it without any modifications.

Problem 2. Duplicate Removal [20 points]

Suppose you have a list of integers $x_0, x_1, \ldots, x_n$. You would like to produce a duplicate-free list, although you're not picky about changing the order of items, so for example if the input is $[1, 3, 5, 3, 2]$ then a perfectly acceptable output is $[5, 2, 3, 1]$.

(a) [10 points] Explain how to do so in O($n \log n$) time in the worst case.

(b) [10 points] Explain how to do so in O(n) time "on average" (or "with high probability", or however else you like to say).

Problem 3. Merging Binary Heaps [20 points]

Suppose you have two binary heaps h_1 and h_2, and you wish to produce a new heap h that combines both of them. That is, h contains all of the items (& their priorities) from both h_1 and h_2. Suppose that h_1 and h_2 have m and n elements, respectively.

(a) [5 points] Explain how to do so in O($n + m$) time. Justify your claim.

(b) [10 points] Suppose $n < m$. Explain how to do so in O($n \log m$) time. Justify.

(c) [5 points] Develop a combined merging strategy that will run in O($\min\{n+m, n \log m, m \log n\}$) time. You can assume the existence of the strategies in (a) and (b), even if you haven't figured them out yourself.

Problem 4. Buggy Code: Exploring Planet Nine [20 points]

Last year, some scientists had an exciting idea: there might exist a ninth planet in our solar system! Planet Nine! Below is some Java code that will certainly not help us to explore Planet Nine. Notice all four excerpts are from the same codebase, and the later parts may refer to code in the earlier parts.

For each excerpt of code, there is a single bug that will either prevent compilation or will cause the program to crash.[1] **Please identify only one bug per part.**

Problem 4.a. [5 points]

```
public class LandingVehicle {

        public String name;
        public int fuel;

        LandingVehicle(){
        }

        public void setName(String n){
                name = n;
        }

        public void dispatch(){
        }

        public boolean testVehicle(int i){
                for (int wheel = 0; wheel < 4; wheel++){
                        testWheel(wheel);
                }
                return true;
        }

        public void testWheel(int w) {
                boolean failed = true;
                for (int i=10; i>0; i--){
                                // Do test.
                }
                if (failed) return (w / i);
        }

}
```

[1]Please do not identify warnings, such as the requirement that each class is in its own file, or that some variables may be unused. Do not identify stylistic problems, such as missing comments or bad variable names. Do not identify problems that cause the computation to produce a different answer than you think it should. Only identify problems that either prevent the program from compiling or cause it to crash.

Problem 4.b. [5 points]

```
    public class Rover {

        public static int roverCount = 0;
        public String pilot;

        public Rover(String p){
            pilot = p;
        }

        public Rover() {
            roverCount++;
        }

        public boolean testVehicle(int i){
            for (int wheel = 0; wheel < 4; wheel++){
                // do test;
            }
            return true;
        }

        public static int analyzeRovers(){
            for (int i=0; i<roverCount; i++){
                if (testVehicle(i)){
                    return -1;
                }
            }
            return 1;
        }
}
```

Problem 4.c. [5 points]

```
public class Probe {

          public String name;

          public Probe(String n){
                  name = n;
          }

          public String checkFuel(){
                  Rover rover = new Rover();
                  if (rover.testVehicle()) return "Ok" else return "Nope!";
          }
}
```

Problem 4.d. [5 points]

```
public class NinthPlanet {

          public LandingVehicle[] rovers;

          public NinthPlanet(String[] names){
                  int numRovers = names.length;
                  rovers = new Rover[numRovers];

                  for (int i=0; i<numRovers; i++){
                           rovers[i].setName(names[i]);
                  }
           }

           public int dispatchRovers() {
                  for (int i=0; i<rovers.length; i++){
                          rovers[i].dispatch();
                  }
                  return 17;
           }
}
```

And now you are ready to launch for Planet Nine!

Problem 5. Linear data structures [20 points]

You are given the `LinkList` class with the following implementation:

```
public class LinkList {
    int num(); // returning the number of elements in the list
    int peekHead(); // return the first element of the list
    int peekTail(); // return the first element of the list
    void prepend(int i); // add the integer i to the head of the list
    void append(int i); // add the integer i to the tail of the list
    void deleteHead(); // delete the first element of the list
    void deleteTail(); // delete the last element of the list
}
```

You can assume all the functions are `public` and well implemented in $O(1)$ time.

Problem 5.a. [10 points] Your job is to write a function `splitIntoTwo(a,b,c)` such that `a,b,c` are linklists. The function will move the first half of `a` into `b` and the second half of `a` into `c`. Thus, after the function, the list `a` will be empty. And the order of the integers in `b` and `c` are the same as when they were in `a`. If `a` has odd number of elements, `b` will have one more element than `c`.

```
void splitIntoTwo(LinkList a, LinkList b, LinkList c)
{

}
```

Problem 5.b. [10 points] You may recall from lecture the "doubly-linked list," and hopefully it has already occurred to you that such a structure would be suitable to implement the `LinkList` class. As described in class, each node contained a "data" field, as well as "next" and "previous" pointers. The front of the list had "previous" equal to null, and the back of the list had "next" also equal to null. The structure as a whole contained two pointers to nodes: "head," which pointed to the first node, and "tail," which pointed to the last.

Please describe how this structure could be modified so that it contained only a single pointer to a node (*e.g.* just "head"). Describe how the operations of `prepend(i)`, `append(i)`, `deleteHead()`, and `deleteTail()` would need to work, paying special attention to any corner cases. Your solutions should still be O(1).

END OF EXAM