General Instructions

1. You may use any printed/written material. Electronic devices are not allowed.

2. The work is to be your own and you are expected to adhere to the UMass Boston honor system.

3. The exam contains five questions. The weight of each question is listed. Read each question carefully before you answer.

4. Write your answers in the available spaces, using the back of the page if needed. Write clearly and concisely and try to avoid cursive.

5. Please explain your answers if needed but do it briefly.

6. If you base your answer on a homework question or class notes state it in your answer.

Good Luck!

Name (as appears on your student ID): ___________________________
1. Runtime analysis

(a) Given the following piece of code:

```java
public static int someFunction(int n)
{
    int i, j, sum = 0;
    for (i=0; i < n; i+=2) /* loop 1 */
        sum += i*i;
    for (i=0; i < n; i++) /* loop 2 */
        for (j=1; j < n; j=j*2) /* loop 3 */
            sum += i*j;
    return sum;
}
```

What is the run time of:

i. Loop 3
   
   \( O(\log n) \) (j is growing in multiples of 2, so after at most \( \log n \) iteration it will reach \( n \))

ii. Loop 2 (including the runtime of loop 3).
   
   Loop 2 runs loop 1 \( n \) times, so overall \( O(n \log n) \)

iii. Loop 1.
   
   \( O(n) \) (notice that while the loop runs \( \frac{n}{2} \) times, the big-O expression is the simplest, no coefficients).

iv. The whole function, explain briefly.
   
   \( O(n \log n) \). First loop 1, followed by loop 2 (3 is nested within 2). So the overall runtime is the sum of both. Notice that the correct expression is \( O(n \log n) \) and not \( O(n + n \log n) \) because we use the simplest form, no coefficients or polynomials, and \( O(n \log n) \) dominates \( O(n) \).

(b) An algorithm takes 15 seconds to solve a problem of size 1000. If the algorithm is quadratic — i.e., runs as \( O(N^2) \), how large a problem can be solved in 60 seconds?

i. 2000
ii. 4000
iii. 6000
iv. none of the above

The answer is 2000. A quadratic algorithm takes four times more when the input size is doubled.

2. Performance with scaling up JDK Collections.

(a) Say you have a HashMap of size 1000 and one search operation takes approximately 1ms. How long will one search take (approximately, on average) on a HashMap of size 2000?

   i. About the same time (since Hash tables are constant).
   
   (b) Same question, only TreeMap.

   This is logarithmic, so very close to constant, probably about 1.1 ms or so.

   (c) Same, but a LinkedList.

   LinkedList search is linear, hence about 2ms.

3. Java Data Structures (from Midterm F16): For (a-c) below, determine what is the best data structure to use out of the ones we discussed in class: List, Set, Map. If more than one acceptable answer exists, use the most efficient one (with respect to runtime) that has the power you need. Also say what type you use: LinkedList vs. ArrayList, HashSet vs. TreeSet or HashMap vs. TreeMap. Please provide an explanation.
(a) You are working on a banking program. Each day, the checks for one account are processed and (assuming they don’t “bounce”) an appropriate Check object is added to the Account object, to wait for end-of-month processing to write the bank statement. The check objects have fields number (of type int), received (of type Date) and amount (of type Money) and must be kept in original order by time of arrival, and are only used once (in our simplified system) to write the statement, where they are reported in the same order.

List<Check>, Set<Check>, Map<Check, String>

Both LinkedList and ArrayList are good. Notice that a TreeSet that compares by date is not good because dates may have duplicates and a List is the simplest way to keep items in the order they were inserted.

(b) In the same banking scenario, each account has a unique id (an Integer) and one or more owners identified by social security numbers (also ints). Also bank account holders may have several accounts, each with a unique id. Explain how you can use two Collections API classes working together to support looking up the bank accounts ids for given social security numbers.

List<Integer>, Set<Integer>, Map<Integer, Set<Integer>>

HashMap and HashSet are best.

(c) You want to be able to display the courses a student took (represented as Strings, like “CS310”) and their grades (represented as Characters between ‘A’ and ‘F’, assume there are no “A-” grades etc.), sorted by the grade. Explain briefly.

List<String>, Set<String>, Map<Character, Set<String>>

TreeMap of TreeSet are good. Letter grades can have duplicates, so just a Map from a Character to a String won’t do. Another option is to make a List<Set<String>> with 5 entries, from A to F (no E, of course).

4. Java implementation: Consider the Queue API from pg. 121:

```java
public class Queue<Item> implements Iterable<Item> {
    Queue() // create an empty queue
    void enqueue(Item item) // add an item
    Item dequeue() // remove the least recently added item
    boolean isEmpty() // is the queue empty?
    int size() // number of items in the queue
}
```

(a) Write a Java interface for the methods of this API, with name QueueIf (If for interface, saving “Queue” for the implementing class)

This is incredibly similar to the code above, just without construction and implementation:

```java
public interface QueueIf<Item> implements Iterable<Item> {
    void enqueue(Item item); // add an item
    Item dequeue(); // remove the least recently added item
    boolean isEmpty(); // is the queue empty?
    int size(); // number of items in the queue
}
```

(b) What can you add to the top line of the source for S&W class Queue of page 151 to say it complies with this interface? Write the new first line for class Queue.

```java
public class Queue<Item> implements QueueIf<Item>
```