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 4 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): _____________________

<table>
<thead>
<tr>
<th>Question</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
1. (30%) Runtime analysis. Given the following piece of code:

```java
public int mystery(int n) {
    int result = 0;
    for (int x = 0; x < n; x++) {
        int y = 1;
        while (y < n)
            y = y * 2;
        result += y;
    }
    return result;
}
```

(a) (6%) What is the run time (Big-Oh) of the (internal) while loop as a function of n? (independently of the external for loop)

(b) (6%) How many times is the external for loop executed as a function of n (exact number!)?

(c) (6%) What is the big-Oh run time of the entire function as the function of n?

(d) (6%) What will the function return for n=4?

(e) (6%) This part is unrelated to (a-d) above. A given algorithm runs as $O(2^n)$ where n is the input size. If the algorithm takes 10 seconds to run on an input of size 5, what is the input size that makes the algorithm run for approximately 40 seconds? Explain briefly.
2. (25%) Java implementation and collections: For each of the following statements say whether it is possible or not using only the API collections we learned in class. Explain each answer briefly.

(a) (5%) A program has set up a `Map` from `String` to `Integer`, i.e. `Map<String, Integer>`. Can it make two different keys, “x” and “y” map to the same `Integer` 6?

(b) (5%) A program has set up a `Map` from `String` to `Integer`. Now it wants to set up the inverse map from `Integer` to `String`. Can it just allocate a new `Map<Integer, String>`, get all the Hash Entries from the original map, reverse the role of key and value and insert it to the new map, such that all the original data is retained?

(c) (5%) The program has set up a List of integers, `ArrayList<Integer>`, and added several elements to it. Now it wants to determine the smallest number in strictly less than $O(n)$ time.

(d) (5%) Same as (c) above only using a `HashSet<Integer>`.

(e) (5%) Same as (c) above only using a `TreeSet<Integer>`.
3. Hash tables (20%): In each one of the following questions provide a brief explanation (no more than a sentence or two).

(a) (7%) Let $H$ be a hash table where collisions are handled by separate chaining. Re-hashing is used whenever the load factor (ratio of items in the table and the size of the table) exceeds $\frac{1}{2}$. Assume that the initial size of $H$ is 2 and re-hashing doubles the size of the table. After inserting 10 items with different keys, what is the size of $H$?

(b) (7%) Consider an initially empty hash table $H$. In the worst case scenario, what is the time complexity (big-Oh) to insert $n$ keys into the table if separate chaining is used to resolve collisions? Suppose that each entry stores a linked list, and when adding a new element to an unordered linked list, such an element is inserted in the beginning of the list. Explain briefly.

(c) (6%) What is the average case run time for (b) above? Explain briefly.
4. Graphs (25%):

(a) (10%) Trace the run of Depth-First search (DFS) algorithm starting from $v_1$ in the graph below. For tracing, use the same notation as in the class notes and HW3 solutions. In case of multiple options follow numerical order. The answer should start with:

Check $v_1$
edge $v_1$−$v_2$

... 

Mark an * near the edges that participate in the final tree. Notice that even though it looks a bit like the graph from HW3, it’s not the exact same graph.

![Graph Diagram]

(b) (10%) This graph is a DAG. provide a topological sorting of the vertices in the graph. Use any algorithm mentioned in the class or HW. State which algorithm you used and outline it briefly. If you use the reverse postorder you may just refer to your results for part (a).

(c) (5%) Does the graph have more than one topological sort? If so – find at least another topological sort, different than (b). Otherwise, explain (Hint: HW).