1. (20%) Runtime analysis. Given the following piece of code:

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

(a) (5%) What is the runtime of the (internal) while loop as a function of n? (independently of the external for loop)

\( O(\log n) \), since we keep multiplying by 2.

(b) (5%) How many times is the external for loop executed as a function of n?

\( n \) times.

(c) (5%) What is the big-Oh run time of the entire function as the function of n? \( O(n \log n) \) since the internal loop runs \( n \) times.

(d) (5%) This part is unrelated to (a-c) 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.

7. In an exponential algorithm adding 1 doubles the runtime. Adding 2 quadruples the runtime.

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?

Yes, values can have duplicates.

(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 HashEntries 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?

No, because if the values had duplicates they will be overridden.

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

No, lists are generally not sorted by value.

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

No, hash tables are generally not sorted by value.

(e) (5%) Same as (c) above only using a TreeSet<Integer>.

Yes, TreeSets are sorted so the minimum can be reached in less than linear time.
3. Hash tables (15%): In each one of the following questions provide a brief explanation (no more than a sentence or two).

(a) (8%) 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 $H$?

32. We double first when there are two elements and then the size is 4. When the third element is inserted we double again to 8. At element 5 we double to 16 and at element 9 we double to 32, where it stays at 10.

(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.

If we don’t consider checking for duplicates the insertion is always $O(1)$, so $n$ insertions are $O(n)$. Notice that if we needed to resolve collisions it would be $O(n^2)$ in the worst case since we would have to go over an entire list of linear size each time.

4. Induction (20%). Prove the following formulas. Don’t forget to explicitly state the three stages:

- Base case
- Inductive hypothesis for $1 < k < n$
- The inductive step from $n - 1 \rightarrow n$ (or $n \rightarrow n + 1$ if it works better)

(a) (10%) Show that $1 + 3 + 5 + \cdots + (2n - 1) = n^2$

- Base: for $n=1$ $1 = 1^2$.
- Assume it’s true for $k = 1 \ldots n - 1$
- Proving for $n$: $\sum_{i=1}^{n} (2i-1) = \sum_{i=1}^{n-1} (2n-1) + 2n-1 = (n-1)^2 + 2n-1$ (by inductive hypothesis) = $n^2 - 2n + 1 + 2n - 1 = n^2$.

(b) (10%) Prove that $4^n - 1$ divides by 3 for all positive integers $n$

- Base: for $n=1$ $4 - 1 = 3$.
- Assume it’s true for $k = 1 \ldots n - 1$
- Proving for $n$: $4^n - 1 = 4 * 4^{n-1} - 1 = 3 * 4^{n-1} + 4^{n-1} - 1$. The first part is a multiple of 3. The second divides by 3 by inductive hypothesis.
5. Graphs (20\%):

(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. 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](image)

The order of visitation is:

<table>
<thead>
<tr>
<th>node/edge</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>Y</td>
</tr>
<tr>
<td>$v_1 - v_2$</td>
<td>Y</td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
</tr>
<tr>
<td>$v_1 - v_4$</td>
<td>Y</td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
</tr>
<tr>
<td>$v_4 - v_2$</td>
<td>N</td>
</tr>
<tr>
<td>$v_1 - v_5$</td>
<td>Y</td>
</tr>
<tr>
<td>$v_5$</td>
<td></td>
</tr>
<tr>
<td>$v_5 - v_3$</td>
<td>Y</td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
</tr>
<tr>
<td>$v_3 - v_2$</td>
<td>N</td>
</tr>
<tr>
<td>$v_3 - v_4$</td>
<td>N</td>
</tr>
</tbody>
</table>

(b) (10\%) This graph is a DAG. Based on your answer to (a) above, provide a topological sorting of the vertices in the graph. Use any algorithm mentioned in the class or HW.

See drawing above for start/end visitation time. You can extract the postorder and reverse it to get $v_1, v_5, v_3, v_4, v_2$. 