Question 1. Suppose that in some class we define two instance variables:

```java
private int[] t = new int[8];
private int count;
```

and that the only code executed to set the values associated with these variables is:

```java
t[0] = 8;
t[1] = 3;
t[2] = 6;
t[3] = 2;
t[4] = 9;
t[5] = 1;
t[6] = 7;
t[7] = 3;
```

```java
count = 5;
```

so that the contents of the array could be described by the table below:

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>3</th>
<th>6</th>
<th>2</th>
<th>9</th>
<th>1</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
</table>

Assume we have defined the method:

```java
public int indexOfSmallest(int size) {
    int posOfMin = 0;
    for (int i = 1; i < size; i++) {
        if (t[i] < t[posOfMin]) {
            posOfMin = i;
        }
    }
    return posOfMin;
}
```

a) What value would be returned by the invocation `indexOfSmallest(count)`?

**Answer:** The value returned would be 3.

**Explanation:** The general purpose of the `indexOfSmallest` method is to find the smallest value in `t` that appears at or before position `size-1`. The if statement within the loop updates the variable `posOfMin` to equal the position where the smallest value at or before position `i` has been found. The for loop varies the value of `i` from 1 to `size - 1`. Therefore, once the loop completes executing, the value left in `posOfMin` (and returned by the method) will be the position of the smallest value in `t` that appears at or before position `size-1`.

b) What value would be returned by the invocation `indexOfSmallest(count-2)`?

**Answer:** The value returned would be 1.

**Explanation:** As described in the explanation for part (a), `indexOfSmallest` looks for the smallest value in `t` at any position up to but not including the value of its parameter `size`. Passing `count - 2` rather than `count` when the method is invoked, eliminates the value 2 found in position 3 of the array from consideration. Therefore, the value 3 found in position 1 because the smallest value among those considered.

c) Consider the following method:

```java
public void mystery( int size ) {
    int remaining = size;

    while ( remaining > 1 ) {
        int pos = indexOfSmallest( remaining );
```
remaining = remaining - 1;
int smallest = t[pos];
t[pos] = t[remaining];
t[remaining] = smallest;

Draw a table showing the values of the elements of t after the execution of the invocation “mystery(count);”.

Answer:

| 9 | 8 | 6 | 3 | 2 | 1 | 7 | 3 |

See part (d) for explanation.

d) Describe the general behavior of the method named mystery.

Answer: Sorts the elements of the array from highest (at index 0) to lowest.

Explanation: The statements that form the body of the loop in the method use indexOfSmallest to find the smallest of the “remaining” elements in t and then interchange the smallest value with the value currently stored at position remaining - 1 in the array.

The loop is designed so that at each repetition the values in the array have two properties: a) any value found in position 0 up to position remaining - 1 will be greater than all of the values stored in any position from remaining through size - 1; b) the values found in positions remaining through size - 1 will be arranged in decreasing numerical order.

Given that these properties are always true when the statements in the loop begin executing, the action of interchanging the smallest element found before position remaining with the element stored at remaining - 1 and then decreasing remaining by 1 ensures that properties (a) and (b) will also always be true each time one execution of the body of the loop is completed.

When the loop finished, remaining will be 0 and property b will still be true so all the numbers from position 0 up to positions size - 1 will be arranged in decreasing numerical order.

Figure 1: An IP Network.
Question 2. Consider the imaginary IP network fragment shown in Figure 1. The network is composed of three physical networks connected to one another by the internal router shown in the center of the diagram. To make it clear that they are distinct physical networks, the diagram shows that each network uses a very distinct networking technology: the two laptops are shown as part of a wireless network, the two cartoon computers are attached to an Ethernet, and the connection between the internal router in the center of the diagram and the external router that connects to the rest of the Internet is drawn as a ring network. The details of ring, wireless and Ethernet technologies, however, are irrelevant to the problem. All that matters is that there are three distinct physical networks involved. Since the internal router has connections to all three physical networks, it has three distinct IP addresses — one for each of the interfaces through which it connects to the three physical networks. The IP addresses associated with router interfaces and with the computers connected to the Ethernet and wireless networks are shown in Figure 1. The physical addresses associated with all of the IP addresses shown in the figure are given in the following table:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Physical Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.134.172.12</td>
<td>02:4F:30:A5:00:1F</td>
</tr>
<tr>
<td>200.134.172.9</td>
<td>39:22:0D:10:3B:33</td>
</tr>
<tr>
<td>137.165.2.1</td>
<td>22:31:FF:21:4B:18</td>
</tr>
<tr>
<td>137.165.4.201</td>
<td>02:4F:30:03:F2:11</td>
</tr>
<tr>
<td>200.134.73.1</td>
<td>39:22:0D:93:83:AA</td>
</tr>
<tr>
<td>200.134.73.19</td>
<td>20:90:33:D0:D6:41</td>
</tr>
<tr>
<td>200.134.73.6</td>
<td>50:57:30:28:01:9B</td>
</tr>
</tbody>
</table>

(a) Any IP packet traveling through this network will include source and destination addresses in the header of the hardware network packet (Ethernet, WiFi, or ring) and source and destination addresses in the IP packet encapsulated in the data area of the hardware packet. Thus, each packet would contain a total of 4 addresses. Would it be possible for a valid packet observed within this collection of networks to contain the addresses 200.134.172.12, 200.134.172.9, 02:4F:30:A5:00:1F and 39:22:0D:10:3B:33? If not, explain why. If so, on which of the three networks shown in the figure (Ethernet, WiFi, or ring) could this packet be observed?

Yes, this is certainly possible. The machines with IP addresses 200.134.172.12 and 200.134.172.9 must be on the same physical networks since their IP address share the common prefix 200.134.172. Therefore, if either of the two machines wanted to send a message to the other, the packet would be sent directly from source to destination without going through any router. The packet could therefore only be observed on the wireless network shown in the diagram. The source and destination IP addresses would be 200.134.172.12 and 200.134.172.9 and the hardware source and destination addresses would be for the same machines: 02:4F:30:A5:00:1F and 39:22:0D:10:3B:33.

(b) Would it be possible for a valid packet observed within this collection of networks to contain the addresses 200.134.172.12, 200.134.73.6, 02:4F:30:A5:00:1F and 50:57:30:28:01:9B? If not, explain why. If so, on which of the three networks shown in the figure could this packet be observed?

No, this is not possible. The IP addresses involved have distinct prefixes (200.134.172 and 200.134.73) indicating that the machines are on separate hardware networks. Any packet sent between the machines associated with these IP addresses would therefore have to travel through the router that interconnects these two addresses. If such a packet were observed on the wireless network, the hardware address of the router’s connection to the wireless network, 01:3F:22:9B:6C:99, would appear in the packet rather than 50:57:30:28:01:9B. If such a packet were observed on the wired Ethernet, the router’s address on the Ethernet, 39:22:0D:93:83:AA, would be found rather than 02:4F:30:A5:00:1F. The four addresses described in the problem statement, would never be observed together.

(c) Would it be possible for a valid packet observed within this collection of networks to contain the addresses 200.134.172.12, 200.134.73.6, 02:4F:30:A5:00:1F and 01:3F:22:9B:6C:99? If not, explain why. If so, on which of the three networks shown in the figure could this packet be observed?
Yes, as described in the answer to the preceding question, these addresses would be found in any packet sent between 200.134.172.12 and 200.134.73.6 while that packet was traveling through the wireless network.

(d) Suppose 200.134.172.12 wants to send an IP packet to 200.134.73.19. How would 200.134.172.12 determine whether the IP packet should be sent to 200.134.73.19 directly or via the router? Once it determined it should send the packet to the router, how would 200.134.172.12 determine the router’s IP address? How would Station 200.134.172.12 determine the router’s physical address?

200.134.172.12 would again begin by determining its own IP network address and the IP network address for 200.134.73.19. Since both IP addresses begin with 200, both machines are attached to class C networks so their network addresses are 200.134.172 and 200.134.73. Since these two network addresses are different, 200.134.172.12 would realize this packet needs to be sent through a router to reach its destination.

200.134.172.12 would then look in its forwarding table to find an entry for network 200.134.73. Chances are, there would be no specific entry for this network, but there would be a default entry indicating that all packets destined to leave network 200.134.172 should be sent to the router 200.134.172.1.

200.134.172.12 would then look in its ARP table for the physical address associated with 200.134.172.1. If the address was not in the ARP table, it would broadcast an ARP request to add the needed information to the ARP table. Once the physical address for 200.134.172.1 was available, it would send the packet to the router.

(e) Suppose that a packet originally sent from the machine with IP address 200.134.73.6 to 202.34.190.4 was observed while traveling through the ring network between the two routers shown in the figure. What would be the IP source and destination addresses and the physical source and destination addresses in the packet?

The source and destination IP addresses will simply be the addresses of the actual source and destinations described: 200.134.73.6 to 202.34.190.4. Neither of the Ethernet addresses, however, will be addresses associated with the actual sources and destinations. While traveling through the ring network, the packet is actually traveling between the two routers. Therefore, its physical source and destination addresses will be the ring network addresses of the two routers: 02:4F:30:03:F2:11 and 22:31:FF:21:4B:18.

Question 3.

Earlier in the semester, we introduced Huffman’s algorithm which finds an optimal binary prefix-free code for a set of $n$ symbols $s_1, \ldots, s_n$ which occur $o_1, \ldots, o_n$ times respectively in some document. The procedure creates a set of trees – one for each symbol – and initializes the weight $w_i$ of each tree to $o_i$. It then selects two trees with minimum weight, and replaces them with a new tree composed of the subtrees. This new tree has weight equal to the sum of weights of its subtrees. This procedure is repeated until only a single tree remains. The resulting tree can be interpreted as a description of a set of codewords for the symbols used.

The cost of a Huffman tree is equivalent to the size of the document when it’s encoded using the code defined by the tree. If symbol $s_i$ appears $o_i$ times and $s_i$ is a leaf at depth $d_i$ in the tree then each occurrence of $s_i$ in the document is encoded with $d_i$ bits. As a result, $s_i$ contributes cost $o_i \times d_i$ to the total cost of the encoded document. In general the total cost of the encoded document (and hence the tree) is

\[ o_1 \times d_1 + o_2 \times d_2 + \cdots + o_n \times d_n \]

(1)

For example, consider the following document (of length 11) composed of 5 symbols:

\begin{center}
ABRACADABRA
\end{center}

The table below lists each symbol and its number of occurrences.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrences</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Running Huffman’s algorithm on the table might produce the following Huffman tree:

![Huffman Tree Diagram]

Encoding the document according to the code defined by the Huffman tree yields

```
10100111000100110100111
```

which has a length of 23. In other words, the cost of the tree is 23. We can verify this is indeed the cost of the code by noting that in the tree, A has depth 1 and the remaining symbols have depth 3. Hence, using just the tree, we can also arrive at the cost by using Equation 1.

\[
1 \times 3 + 1 \times 3 + 2 \times 3 + 2 \times 3 + 5 \times 1 = 3 + 3 + 6 + 6 + 5 = 23
\]

Surprisingly, one does not need to actually create a tree to compute the cost of a Huffman code. To see how, imagine that we are creating the Huffman tree shown above. Initially, our running total is \( T = 0 \). Every time we merge two trees, the codewords for the symbols in those trees increase by 1 bit. Our procedure works by adding in this additional codeword cost at the time of a merge.

- We begin by merging C and D. We now know that each occurrence of C and D will be replaced by codewords with length at least 1, so each occurrence of C and each occurrence of D will contribute cost at least 1 to the overall cost. Since C and D each occur only once, \( T = T + 2 = 0 + 2 = 2 \).
- Next we merge B and R. Since B and R will be replaced by codewords with length at least 1, each occurrence of B and each occurrence of R will contribute cost at least 1 to the overall cost. Since B and R each occur twice they will contribute an additional cost of \( 2 + 2 = 4 \) to \( T \). Hence, \( T = 6 \).
- Next we merge the CD tree with the BR tree. The CD tree has weight 2. This means C and D occur 2 times collectively in the document. The BR tree has weight 4. This means B and R occur 4 times collectively in the document. Since we are merging these two nodes, we know the codewords for C and D will be 1 bit longer in the document. The same is true for B and R. Since C and D occur twice collectively, this merge adds a cost of 2 to the total. Similarly, since B and R occur 4 times collectively, the merge adds a cost of 4 to the total. Hence the total is now \( T = T + 2 + 4 = 6 + 2 + 4 = 12 \).
- Finally, we merge the CDBR node with the A node. Each occurrence of A will be encoded with a codeword of length 1 and the symbols in the CDBR node each require an additional bit in their keywords. Since A occurs 5 times and C,D,B, and R occur 6 times collectively, our total is \( T = T + 5 + 6 = 12 + 5 + 6 = 23 \).

Note that while we discussed computing the running total in terms of tree merges, we don’t actually need to create or merge any trees; we just need to find the two smallest weights and replace them with their sum in the current list of weights. This means each merge operation is really just a sum operation. Each sum operation has an associated update to the running total so it makes sense to talk about the value of the running total after each sum operation.

The series of steps that would be performed when applying this tree-less version of the algorithm to the occurrence counts for the ABRACADABRA example are shown below. In each step, the value in the list of occurrence counts that has been produced by merging two of the smallest occurrence counts in the previous list is shown in bold face. It is this value that is added to “Running total” at each step.
Consider a document using 8 symbols whose occurrence counts are shown in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrences</td>
<td>2</td>
<td>5</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>24</td>
<td>2</td>
</tr>
</tbody>
</table>

Fill in the cells in the tables provided below to show how the algorithm described above would unfold when applied to a message with these occurrence counts. To help you know whether you did this correctly, we have filled in the final values.

<table>
<thead>
<tr>
<th>occurrences</th>
<th>2</th>
<th>5</th>
<th>20</th>
<th>7</th>
<th>3</th>
<th>2</th>
<th>24</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>4</th>
<th>5</th>
<th>20</th>
<th>7</th>
<th>3</th>
<th>24</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>4</th>
<th>5</th>
<th>20</th>
<th>7</th>
<th>5</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>9</th>
<th>20</th>
<th>7</th>
<th>5</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>9</th>
<th>20</th>
<th>12</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>21</th>
<th>20</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>51</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>41</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>92</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occurrences</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running total</td>
<td>157</td>
</tr>
</tbody>
</table>