Question I

Part (a): Let $X_1$ and $X_2$ denote the RVs that represent the values obtained in the two trials. We need to compute $\mathbb{E}[X_1 + X_2]$, which by linearity is equal to $\mathbb{E}[X_1] + \mathbb{E}[X_2]$.

Since $X_1$ and $X_2$ have the same probability distribution and all the $n$ values in $A$ are equally likely,

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = \sum_{i=1}^{n} i \Pr\{X_1 = i\} = \frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \times n(n+1)/2 = \frac{n+1}{2}.$$  

Hence, $\mathbb{E}[X_1 + X_2] = n + 1$, as required.

Part (b): $X$ is a binomial RV since it represents the number of successes in $n = 25$ trials. Since the success of a trial is determined by two of the 10 values in $B$, the success probability $p$ for each trial $= 2/10 = 1/5$. The variance of a binomial RV is given by $np(1-p)$. Thus, $\text{Var}[X] = 25 \times (1/5) \times (4/5) = 4$.

Part (c): The last candidate will be hired if and only if the rank is $n$. Of all the $n!$ permutations, the number of permutations in which the last candidate has the highest rank = $(n-1)!$. Therefore, the probability that the last candidate will be hired $= (n-1)!/n! = 1/n$. Hence, the probability that the last candidate won't be hired $= 1 - (1/n)$.

Question II

Part (a): The array $B$ after 105 is inserted is as follows:

<table>
<thead>
<tr>
<th>i</th>
<th>105</th>
<th>100</th>
<th>80</th>
<th>90</th>
<th>60</th>
<th>50</th>
<th>40</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>B[1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B[2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B[3]</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B[5]</td>
<td>60</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B[6]</td>
<td>40</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part (b): We know that the nodes in positions $[n/2] + 1$ through $n$ of a binary heap are leaves. (One can see this by noting that the left child of the node in position $[n/2] + 1$ is outside the heap.) Thus, there are at least $n - ([n/2] + 1) + 1 = n - [n/2] = [n/2]$ leaves.

To see that there are no other leaves, we observe that the node in position $[n/2]$ is not a leaf; its left child is at position $2 \times [n/2] \leq n$, which is within the heap. Thus, none of the nodes in positions 1 through $[n/2]$ is a leaf. That is, the number of leaves is exactly $[n/2]$.

Part (c): We will prove this by contradiction. Suppose it is possible to implement any sequence of $2n$ INSERT and EXTRACT-MAX operations in time $f(n) = o(n \log n)$ in the worst-case. Consider the following sequence of $2n$ operations for sorting any set $S$ of $n$ numbers:

1. Insert the elements of $S$ one by one into the Max-Heap. (This leads to $n$ INSERT operations.)
(2) Carry out \( n \) Extract-Max operations on the heap.

Thus, we have a comparison-based sorting algorithm which runs in time \( f(n) = o(n \log n) \). This violates the known lower bound on sorting. Hence, the worst-case time for the \( 2n \) operations is \( \Omega(n \log n) \). ■

---

**Question III**

**Part (a):** (Note that columns must be sorted from right to left.)

<table>
<thead>
<tr>
<th>RAT</th>
<th>RIG</th>
<th>WAR</th>
<th>ASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>ASK</td>
<td>RAT</td>
<td>CAT</td>
</tr>
<tr>
<td>WAR</td>
<td>----&gt;</td>
<td>WAR</td>
<td>----&gt;</td>
</tr>
<tr>
<td>ASK</td>
<td>RAT</td>
<td>RIG</td>
<td>RAT</td>
</tr>
<tr>
<td>RIG</td>
<td>CAT</td>
<td>PIT</td>
<td>RIG</td>
</tr>
<tr>
<td>PIT</td>
<td>PIT</td>
<td>ASK</td>
<td>WAR</td>
</tr>
</tbody>
</table>

**Part (b):** Steps 1 and 3 of Bucket-Sort run in \( O(n) \) time. So, we focus on the analysis of Step 2.

Let bucket \( B[j] \) contain \( N_j \) keys, \( 0 \leq j \leq n - 1 \). The time used to sort \( B[j] \) is at most \( cN_j^2 \), because of the use of Insertion-Sort. Therefore, the total time \( T(n) \) to sort all the buckets is bounded by

\[
T(n) \leq c \sum_{j=0}^{n-1} N_j^2
\]  

To get an upper bound on the right side of Equation (1), we use the following fact.

**Fact:** If \( N_0, N_1, \ldots N_{j-1} \) are nonnegative integers, then \( \sum_{j=0}^{n-1} N_j^2 \leq \left( \sum_{j=0}^{n-1} N_j \right)^2 \).

(Proof of Fact: Note that)

\[
\left( \sum_{j=0}^{n-1} N_j \right)^2 = \sum_{j=0}^{n-1} N_j^2 + 2 \sum_{0 \leq k < j \leq n-1} N_k N_j
\]

The given fact follows since each of the product terms \( N_k N_j \) is nonnegative.)

Thus, from the above fact, we get

\[
T(n) \leq c \left( \sum_{j=0}^{n-1} N_j \right)^2
\]

Now, since \( \sum_{j=0}^{n-1} N_j = n \), we see that \( T(n) \leq cn^2 \). Thus, Step 2 runs in \( O(n^2) \) time. Since Step 2 dominates the running time, the overall running time of the algorithm is also \( O(n^2) \). ■
Question IV:

**Idea:** For each row $i$, we can use binary search to find the largest column number $j$ such that $A[i,j] = 1$. Assuming that column indices vary from 1 to $n$, the value of $j$ gives the number of 1’s in row $i$. This uses only $O(\log n)$ time per row and hence only $O(n \log n)$ total time. The complete algorithm is as follows.

1. total = 0.

2. for $i = 1$ to $n$ do
   1.1. Use binary search on row $i$ to find the largest column $j$ such that $A[i,j] = 1$.
   1.2. total = total + $j$

3. Print total.

**Running time analysis:** Steps 1 and 3 run in $O(1)$ time. In Step 2, the loop executes $n$ times. Each iteration of the loop executes in $O(\log n)$ time, since it involves a binary search of the $n$ elements of a row. So, Step 2, and hence the whole algorithm, runs in $O(n \log n)$ time.