1. You have just finished sorting an array $A[1..n]$ of $n$ distinct numbers into increasing order. When you go out to have a break, your mischievous friend, John, has divided your array into two parts $A_{left} = A[1..i]$ and $A_{right} = A[i+1..n]$, and re-arrange the array so that he puts $A_{right}$ in front of $A_{left}$; precisely, the array now becomes $A[i+1..n]A[1..i]$. See Figure 1 for an example.

After you come back, John tells you about what he has done, but without telling you the value of $i$. To reverse the change, you want to locate the entry with the minimum item, as this will be the boundary between $A_{right}$ and $A_{left}$.

(a) (15%) Design an $O(\log n)$-time algorithm to find the position of the minimum item. 
(b) (15%) Show that your algorithm is correct.

2. Consider the following code `ComputeCount`:

```plaintext
ComputeCount()
    1. Input a positive integer $n$;
    2. Set $count = 0$;
    2. for $j = 1, 2, \ldots, n$
    3. if $j$ is a factor of $n$
    4. { Update $count$ to become $1 - count$; }
    5. Output $count$;
```

(a) (15%) The above code computes the value of $count$ in $\Theta(n)$ time. Design a faster algorithm that can compute $count$, and analyze its running time.

- For this problem, the marks will also depend on the quality of your algorithm. 
  At most 15% if your algorithm runs in $O(\log n)$ time; otherwise, at most 5% if it runs in $o(n)$ time, and 0% if it runs in $\Theta(n)$ time.

(b) (15%) Explain why your algorithm is correct.
3. The \textbf{BubbleSort} algorithm is a very simple algorithm for sorting an array of numbers. Given an input array $A[1..n]$ with $n$ distinct numbers, \textbf{BubbleSort} works by repeatedly swapping adjacent elements in $A$ as follows:

\begin{verbatim}
BubbleSort(A)
1. for Phase $k = 1, 2, \ldots, n$
2. for Position $j = 1, 2, \ldots, n - 1$
4. { Swap the entries $A[j]$ and $A[j + 1]$; }
\end{verbatim}

(a) (15\%) Show that \textbf{BubbleSort} is correct.


- For example, if the array is $(2, 3, 6, 4, 0)$, then the pair $(3, 0)$ is inverted, and in total there are 5 inverted pairs.

(15\%) Show that the number of inverted pairs in $A$ is exactly equal to the number of swaps when we perform \textbf{BubbleSort} on $A$.

** (c) (10\%) By using brute force approach, one can easily count the number of inverted pairs of $A$ in $\Theta(n^2)$ time. Design an algorithm that counts the number of inverted pairs in $O(n \log n)$ time.

** \textit{Q3(c) is the hardest question. Spend more time and try your best to solve it!}

4. (No marks.) Give asymptotic upper bound for $T(n)$ in each of the following recurrence. Make your bounds as tight as possible.

(a) $T(n) = 9 \ T(n/2) + n^3$
(b) $T(n) = 7 \ T(n/2) + n^3$
(c) $T(n) = T(\sqrt{n}) + \log n$
(d) $T(n) = 0.5 \ T(n/2) + n$
(e) $T(n) = 3 \ T(n/3) + n/3$