1. John, after taking the lecture on asymptotic notation, has tried to prove that
   \[ 1 + 2 + \cdots + n = O(n) \]
   His proof is by induction, where he attempts to show \( 1 + 2 + \cdots + i = O(n) \) for all \( i \geq 1 \).
   The following is his proof:
   
   1. First, \( 1 = O(n) \), so that the base case \( i = 1 \) is true.
   2. Assume that the statement is true for all \( i = 1, 2, \ldots, k \).
   3. Then, by the above assumption, we have
      \[ 1 + 2 + \cdots + k + (k + 1) = O(n) + (k + 1) \]
      Since \( O(n) + (k + 1) = O(n) \), we have
      \[ 1 + 2 + \cdots + k + (k + 1) = O(n) \]
      thus showing the inductive case is correct.
   4. By mathematical induction, the statement is true for all \( i \geq 1 \), so that
      \[ 1 + 2 + \cdots + n = O(n) \]
   (25%) Obviously, you know for sure that \( 1 + 2 + \cdots + n = n(n + 1)/2 = \Theta(n^2) \), so that there must be something wrong in John’s proof. Can you find the error?

2. In the lecture, we have seen that \textbf{insert} operation in a heap \( T \) can be done as follows:
   
   1. Construct a node \( \ell \) storing the new number;
   2. Add \( \ell \) as a leaf in \( T \), such that after the modification, \( T \) will still satisfy the shape property;
   3. Set node \( x = \ell \);
   4. \textbf{while} ( \( x \) is not root and number in \( x \) ≤ number in parent of \( x \) )
      
      \{ 
      \hspace{1cm} Swap the numbers in \( x \) and in parent of \( x \);
      \hspace{1cm} Update \( x \) to become parent of \( x \);
      \} 
   
   (25%) Show that the above procedure correctly restores the heap property.

3. Peter has given you an array \( A \) of \( n \) distinct numbers, and he wants you to sort \( A \) for him. Further, Peter has informed you that the array is \textit{nearly} sorted: for each number, its
current position (in $A$) and its correct position (when sorted) differ by at most $d$ positions. Precisely, the $k$th smallest number is now stored at $A[j]$ with $k - d \leq j \leq k + d$. See Figure 1 for an example of a nearly sorted array when $d = 3$.

(a) (25%) Give an $O(n \log d)$-time algorithm to sort $A$.
(b) (25%) Show that your algorithm is correct.