The actual midterm will be closed book and closed notes. Since this is a practice exam, and since you will presumably be practicing with open book, it is slightly harder. Show your work for partial credit. If you need extra space, write on the back, and indicate on the front where the rest of the solution is.

Problem 1 (20 points)

Consider the following procedure Select which takes as input an array $A$ of integers, the length $n$ of $A$ and an integer $k$ in the range $1 \leq k \leq n$. Select permutes the elements of $A$ so that the last $k$ entries $A$ are the $k^{th}$ largest elements. This is accomplished by the following code. Select calls the procedure Build-Heap and Heapify.

Procedure Select($A, n, k$)
1. Build-heap($A, n$)
3. For $i \leftarrow n - 1$ downto $n - k + 1$ do
4. Heapify($A, 1$)
6. End For $i$
7. Return

Show that Select takes time $O(n + k \log(n))$.

Problem 2 (25 pts)

Consider the following binary insertion sorting algorithm:

1. For $i \leftarrow 2$ to $n$ do
2. $x \leftarrow A[i], l \leftarrow 1, r \leftarrow i - 1$
3. While $l \leq r$ do
4. $m \leftarrow (l + r)/2$
5. If $x < A[m]$ then $r \leftarrow m - 1$ else $l \leftarrow m + 1$
6. For $j \leftarrow i - 1$ downto $l$ do
8. $A[l] \leftarrow x$

Show that if $C(n)$ is the number of comparisons and $S(n)$ is the number of swaps that $C(n) = O(n \log(n))$ and $S(n) = O(n^2)$. What would be a worst case input for maximizing $C(n)$? Similarly for $S(n)$?
Problem 3 (25 pts) Assume the recurrence \( T(n) = T(n-1) + T(n-2) \), with \( T(0) = 2 \) and \( T(1) = 4 \). Prove by induction that \( T(n) = O(2^n) \).

Problem 4 (10 points) If \( f(n) = \Theta(g(n)) \), then indicate whether the following are true or false for all positive functions \( f(n) \) and \( g(n) \):

(i) \( f(n) = O(g(n)) \)

(ii) \( f(n) = o(g(n)) \)

(iii) \( f(n) = \Omega(g(n)) \)

(iv) \( f(n) = \omega(g(n)) \)

Problem 5 (50 points) Give pseudo-code to merge two heaps into a single heap in \( O(n) \) time. (Hint: Recall \texttt{Heap-Extract-Max} (given below). Create a certain binary tree based on the two input heaps in \( O(n) \) time, and then use logic similar to that of \texttt{Heap-Extract-Max}.)

\texttt{Heap-Extract-Max(A)}

1 \textbf{if} \ heap-size\[A]\ < \ 1
2 \textbf{then} \textbf{error} “heap underflow”
3 \texttt{max}−− \ A[1]
4 \( A[1] \leftarrow A[\text{heap-size}[A]] \)
5 \( \text{heap-size}[A] \leftarrow \text{heap-size}[A] - 1 \)
6 \texttt{HEAPIFY}(A,1)
7 \texttt{return} \texttt{max}
Problem 5. (20 points)

Suppose a person tells you he’s thinking of an integer between 1 and 10, and asks you
to guess it. If you guess right, the game ends. If you guess wrong, he will tell you if the
number is larger or smaller and ask you to guess again. You’ve played this game with this
guy before, and you know the probability, \( p_i \), that he is thinking about \( i \). \( \sum_{1 \leq j \leq 10} p_j = 1 \)
Describe how to use dynamic programming so as to minimize the average number of guesses
needed during a game. Include equations, and define any variables you use.