Exam 1

Write your answers on the exam paper. Use the back if necessary, but label all answers. Read all of the questions carefully. Good luck!

1. Consider the function \( f(n) = n^2 + n \). Show that \( f(n) = O(n^2) \). Is \( f(n) = o(n^2) \)? Why or why not?

2. (15 points) Consider the function \( f(n) = n^2 + n \). Show that \( f(n) = O(n^2) \). Is \( f(n) = \Omega(n^2) \)? Why or why not? Is \( f(n) = o(n^2) \)? Why or why not?

3. Consider the following heap array:

\[
\begin{array}{ccccccccccc}
18 & 12 & 15 & 9 & 11 & 14 & 8 & 7 & 6 & 10 & 4 & 13 & 5 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13
\end{array}
\]

a. (10 points) Show the binary tree form for this heap.

b. (7 points) Is this tree a max-heap? Why or why not?
4. Consider the hiring problem from Chapter 5. Give your reasons for your answers to the questions below for full credit. Think carefully about how candidates are hired.
   a. *(6 points)* What is the probability that the algorithm hires only the first two candidates from a random list of candidates?

5. *(4 points)* What is the probability that the algorithm hires only the first candidate seen and the last candidate seen from a random list of candidates. Consider the following code for the Bubblesort algorithm:

```
BUBBLESORT(A)
1. for i ← 1 to length[A]
2.   do for j ← length[A] downto i + 1
3.     do if a[j] < A[j - 1]
```
   a. What is a loop invariant which must for at the beginning of each iteration of the for loop on lines 2-4?
   b. What is the worst case running time of Bubblesort? What is the best case running time of Bubblesort?

6. Consider the hiring assistant problem from Chapter 5. What is the probability that the algorithm hires the first and last candidates from a random list of candidates?

7. Consider the following heap array:

```
18 12 15 9 11 14 8 7 6 10 4 13 5
```
   a. Show the binary tree form for this heap.
   b. Is this tree a max-heap?

8. A binary search operates over a sorted array. It compares the key, \( v \), to the middle entry of the array. If there is a match, the middle value is returned. If the key is greater than the middle value, binary search is called again with the top half of the array. Otherwise, it is called with the lower half of the array. The pseudocode for a recursive binary search is shown below:

```
RECURSIVE-BINARY-SEARCH(A, v, low, high)
if low > high then return NIL
mid ← ⌊(low + high) / 2⌋
if v = A[mid] then return mid
if v > a[mid] then return RECURSIVE-BINARY-SEARCH(A, v, mid + 1, high)
else return RECURSIVE-BINARY-SEARCH(A, v, low, mid - 1)
```
   a. Write a recurrence equation for binary search, treating it as a divide and conquer algorithm.
b. Solve the recurrence equation to get the runtime for binary-recursive search.

9. Consider the following recurrence equation: \( T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/3) + 1 & \text{otherwise} \end{cases} \). Show the recursion tree for this recurrence equation. Indicate the height of the tree, the number of leaves, and the cost of each node. Also, indicate the cost of each level of the recursion tree.

10. **(20 points)** Consider the list of numbers:

as a max-heap priority queue. Show how to remove the maximum element, leaving the remaining items as a max-heap priority queue.