You will submit problems 1,2,3 and problems 4,5 in separate stapled sets. Print and fill out two copies of this sheet; attach a copy to the top of each of the two parts of your homework. Indicate which part is which by circling the relevant problems [1,2,3 or 4,5] on the cover sheet. On both copies of the cover sheet, neatly print your name(s), NetID(s), and the alias(es) you used for Homework 0 in the boxes above.
Required Problems

1. **The Hard Life of a Journalist**
   **[20 Points]**

(a) **[2 Points]** A journalist, named Jane Austen, travels to Afghanistan, and unfortunately falls into the hands of Bin Laden. Bin Laden offers Jane a game for her life – if she wins she can leave.

The game board is made out of 2 × 2 coins:

```
   H  T
   T  H
```

At each round, Jane can decide to flip one or two coins, by specifying which coins she is flipping (for example, flip the left bottom coin, and the right top coin); next, Bin Laden goes and rotates the board by either 90, 180, 270, or 0 degrees (of course, rotation by 0 degrees is just keeping the coins in their current configuration).

The game is over when all the four coins are either all heads or all tails. To make things interesting, Jane does not see the board, and does not know the starting configuration. Describe an algorithm that Jane can deploy, so that she always wins. How many rounds are required by your algorithm?

(b) **[5 Points]**

After escaping from Bin Laden, and on her way to Kabul, Jane meets a peace loving, nuclear reactor selling, French diplomat. The French diplomat is outraged to hear that Jane prefers Hummus to French Fries, and instructs his bodyguards to arrest Jane immediately, accusing her of being a quisling of the French cuisine (Jane has French citizenship). Again, the diplomat offers her a game for her life, similar to the Bin Laden game, with the following twist: after Jane flips her coins, the diplomat will reorder the coins in an arbitrary order (without flipping any coin). Describe an algorithm that Jane can use to win the game. What is the expected number of rounds Jane has to play before winning (the lower your bound, the better)?

(c) **[5 Points]** After escaping from the French diplomat, Jane travels to Hanoi to investigate rumors that the priests in charge of the Towers of Hanoi games are spending all the money they get on buying computer games and playing them, instead of playing the holy game of Towers of Hanoi, as they are supposed to do.

However, the head priest is willing to do an interview with Jane, only if she plays the coin game (using the French diplomat version), with n coins. Describe an algorithm
that guarantees that Jane wins. Provide an upper bound (as tight as possible) on the number of rounds Jane has to play before winning. (Providing an exact bound here is probably hard. As such, a rough upper bound would be acceptable.)

(d) [5 Points] Jane, tired of all those coin games, goes to Nashville for a vacation. Unfortunately for her, she is kidnapped by an Elvis lookalike. Not surprisingly, he offers her to play the coin game for her life, with the following variants: There are $n$ coins, and at each round Jane can choose which of the $n$ coins she wants to flip. Before flipping the coin, the Elvis lookalike tells her whether the coin is currently head or tail, and Jane can decide whether she wants to flip this coin or not. After each round, the Elvis lookalike takes the coins and reorders them in any order he likes. Describe an algorithm that guarantees that Jane wins. Provide an exact bound on the expected number of rounds that Jane has to play before she wins. (The smaller your bound, the better.)

2. Sorting Random Input

[20 Points]

Let $a_1,\ldots,a_n$ be $n$ real numbers chosen independently and uniformly from the range $[0,1]$.

- [5 Points] Describe an algorithm with an expected linear running time that sorts the numbers.
- [5 Points] Show that the linear running time is with high probability.

- Approx Max Cut

[10 Points]

Given a graph $G = (V,E)$ with $n$ vertices and $m$ edges, describe an algorithm that runs in $O(n)$ time, and outputs a cut $S \subseteq V$, such that the expected number of edges in the cut is $\geq M/2$, where $M$ is the number of edges in the maximum cut, where the number of edges in the cut is $|(S \times (V \setminus S)) \cap E|$.

3. Random Bits in a Treap

[20 Points]

Let’s analyze the number of random bits needed to implement the operations of a treap. Suppose we pick a priority $p_i$ at random from the unit interval. Then the binary representation of each $p_i$ can be generated as a potentially infinite series of bits that are the outcome of unbiased coin flips. The idea is to generate only as many bits in this sequence as is necessary for resolving comparisons between different priorities. Suppose we have only generated some prefixes of the binary representations of the priorities of the elements in the treap $T$. Now, while inserting an item $y$, we compare its priority $p_y$ to the other’s priorities to determine how $y$ should be rotated. While comparing $p_y$ to some $p_i$, if their current partial binary representation can resolve the comparison, then we are done. Otherwise, the have the same partial binary representations (upto the length of the shorter of the two) and we keep generating more bits for each until they first differ.

(a) Compute a tight upper bound on the expected number of coin flips or random bits needed for a single priority comparison. (Note that during insertion, every time we decide whether or not to perform a rotation, we perform a priority comparison. We are interested in the number of bits generated in such a single comparison.)
(b) Generating bits one at a time like this is probably a bad idea in practice. Give a more practical scheme that generates the priorities in advance, using a small number of random bits, given an upper bound $n$ on the treap size. Describe a scheme that works correctly with probability $\geq 1 - n^{-c}$, where $c$ is a prespecified constant.

4. A Game of Death
[20 Points]

Death knocks on your door one cold blustery morning and challenges you to a game. Death knows that you are an algorithms student, so instead of the traditional game of chess, Death presents you with a complete binary tree with $4^n$ leaves, each colored either black or white. There is a token at the root of the tree. To play the game, you and Death will take turns moving the token from its current node to one of its children. The game will end after $2n$ moves, when the token lands on a leaf. If the final leaf is black, you die; if it’s white, you will live forever. You move first, so Death gets the last turn.

You can decide whether it’s worth playing or not as follows. Imagine that the nodes at even levels (where it’s your turn) are OR gates, the nodes at odd levels (where it’s Death’s turn) are AND gates. Each gate gets its input from its children and passes its output to its parent. White and black stand for TRUE and FALSE. If the output at the top of the tree is TRUE, then you can win and live forever! If the output at the top of the tree is FALSE, you should challenge Death to a game of Twister instead.

(a) [10 Points] Describe and analyze a deterministic algorithm to determine whether or not you can win. [Hint: This is easy!]

(b) [10 Points] Unfortunately, Death won’t let you even look at every node in the tree. Describe a randomized algorithm that determines whether you can win in $\Theta(3^n)$ expected time. [Hint: Consider the case $n = 1$.]

5. Closest Numbers
[20 Points]

Let $P$ be a set of $n$ real numbers. The purpose of this exercise is to develop a linear time algorithm for deciding whether there are two equal numbers in $P$. Let $x_1, \ldots, x_n$ be a random permutation of the numbers in $P$. 
(a) [5 Points] Let $\pi_i = \min_{1 \leq k < j \leq i} |x_k - x_j|$ be the distance between the closest pair of numbers in $x_1, \ldots, x_i$. Prove that $\Pr[\pi_i \neq \pi_{i-1}] \leq 2/i$.

(b) [5 Points] Given a parameter $r$, describe an algorithm that decides, in $O(i)$ time, whether $\pi_i < r$. Furthermore, if $\pi_{i-1} = r$ but $\pi_i < r$, then it computes $\pi_i$. (Hint: use hashing and the floor function.)

(c) [5 Points] Show how to modify the previous algorithm into a data-structure, so that after computing $\pi_i$, one can insert $x_{i+1}, \ldots, x_j$ into the data-structure in $O(1)$ time per element, where $\pi_{i+1} = \pi_{i+2} = \cdots = \pi_{j-1} > \pi_j$. And furthermore, the data-structure returns $\pi_j$.

(d) [5 Points] Describe an algorithm, with $O(n)$ expected running time, that computes $\pi_n$. Clearly, if $\pi_n = 0$ then there are two identical numbers in $P$. (Hint: Use (a) and (c)).

(Note that the algorithm of (d) is faster than one can achieve in the comparison model (i.e., we only get to compare numbers). One can prove that the fastest algorithm for this problem in the comparison model requires $\Omega(n \log n)$ time. Namely, the only way to solve it is using sorting.)