Answer the questions below. For the questions involving both programming and analysis, please submit your code through Udacity and your analysis through T-square.

1. In a list of distinct numbers $a_1, \ldots a_n$, we say that the elements a_i and a_j are *inverted* if $i < j$ but $a_i > a_j$. Suppose that all orderings of $a_1, \ldots a_n$ are equally probable under some probability distribution (In other words we shuffled a set of numbers perfectly to obtain the order $a_1 \ldots a_n$.) What is the expected number of inversions?

2. Consider the following greedy algorithm for finding a matching.
   - Initialize S to an empty set of edges
   - While there is an edge where *both* vertices are unmatched by S
     - Add the above edge to S
   - Return S

   a. What is the running time of this algorithm?
   b. Give an example of a graph where the algorithm is not guaranteed to find the maximum matching.
   c. Show that the matching found by the algorithm always has at least half as many edges as a maximum matching.
   d. (Bonus) Now consider a similar algorithm for finding a maximum weight matching in an edge-weighted graph: Greedily add the heaviest edge possible to the current matching; stop when no further edge can be added. Show that this algorithm finds a matching whose weight is at least half the optimum.

3. Consider the following factor 2 approximation algorithm for the minimum cardinality vertex cover problem. Find a depth first search tree in the given graph, G, and output the set, say S, of all the non-leaf vertices of this tree. Show that S is indeed a vertex cover for G and $|S| \leq 2 \cdot OPT$.

   Hint: Use the tree to construct a matching that includes all the vertices in S.

4. The following is known as the maximum acyclic subgraph problem. Given a directed graph $G = (V, E)$, pick a maximum cardinality set of edges from E so that the resulting subgraph is acyclic. Implement an algorithm that gives a $1/2$ factor approximation here.

   https://www.udacity.com/course/viewer#!/c-ud557/l-1209378918/m-3519618629

   Hint: Rather than thinking about picking $1/2$ the optimum, think about picking $1/2$ of the edges.

5. In the lesson, we gave a randomized algorithm for finding a minimum cut set in graph. Now, we consider the problem of finding a *maximum* cut set (all weights on edges are all one).
    a. Consider an algorithm that partitions the vertices into two sets A and B by placing each vertex uniformly and independently at random into one of the two sets. What is the expected value of $C(A,B)$, i.e. the number of edges crossing the cut?
    b. Use the method of conditional expectation to derandomize the above algorithm so that it always achieves this expected value. Implement your algorithm here.

    https://www.udacity.com/course/viewer#!/c-ud557/l-1209378918/m-3481888754

    c. Explain why your algorithm is correct. Use the notation $x_k = A$ to describe the event that vertex k is assigned to the set A in the partition. Use Y to denote the number of cut edges, and let $P : \{1, \ldots n\} \rightarrow \{A, B\}$ represent the partition returned by the derandomized algorithm.