Algorithms I, Supervision 2 - Algorithm Design
Participants ________________

1. ________________________________
2. ________________________________
3. ________________________________
4. ________________________________

1 Algorithm Design

1.1 Dynamic Programming

1. test understanding: how to minimize number of scalar multiplications when finding the product matrix
2. common types of subproblems
   (a) Optimal substructure
   (b) Overlapping subproblems
3. longest increasing subsequence
4. shortest edit distance
5. 0-1 knapsack - use cases

1.2 Greedy algorithms

1. prove that a locally optimal solution (decision) yelds a globally optimal solution
2. give change with minimum number of bancnotes (unlimited supply)
3. related: change of base - from decimal to binary
4. a lot of graph algorithms - (will be studied in Algorithms II)

1.3 Backtracking

1. chess: n queens that don't atack eachother
2. output all permutations/combinations

1.4 Others

1. Divide Conquer (seen)
2. Million Monkeys (MM)
3. Heuristics, state space lower bounds (never overestimate!) - the 15 puzzle problem

Algorithm Design Workset[1]

Murray Edwards: Due on 14th May, 12:30.
Queens, Robinson: Due on 16th May, at the same time as that of your Thursday supervision

All theoretical exercises - mandatory. Besides those, pick at least one of the implementation challenges.

1. **Challenge (optional):** Implement a small diff algorithm using an approach similar to that used in finding the shortest edit distance. There are faster algorithms for finding the diff between two files. Focus on diffing two sentences. How could you extend this to entire files?
2. Explain in what way is the problem of optimal parenthesization of a matrix chain product equivalent to the problem of finding the optimal triangulation of a convex polygon (Use CLRS as a reference, and try to give your own explanation.)
3. www.cs.berkeley.edu/~vazirani/algorithms/chap6.pdf. Solve problems: 6.4, 6.13, 6.23
4. Prove that a greedy strategy is correct for the problem of giving change with the minimum number of bancnotes.
5. **Implementation:** Write a program that prints all the k-combinations of a given set of n elements. (k,n, and the set are given as inputs)

[1]Late submissions will only be accepted on special circumstances, and have to be announced at least one day before the deadline, by email at lc525@cam.ac.uk