- The instructions are the same as in Homework-0, 1, 2, 3.

There are 5 questions for a total of 100 points.

---

1. ~~*(Please note that this question will be counted towards Homework-3.)*~~

   Consider two binary strings $x = x_1, ...x_n$ and $y = y_1, ..., y_m$. An interleaving of x and y is a strings $z = z_1, ..., z_{n+m}$ so that the bit positions of z can be partitioned into two disjoint sets X and Y so that looking only at the positions in X, the sub-sequence of z produced is x and looking only at the positions of Y, the sub-sequence is y. For example, if $x = 1010$ and $y = 0011$, then $z = 10001101$ is an interleaving because the odd positions of z form x, and the even positions form y. The problem is: given x, y and z, determine whether z is an interleaving of x and y.

   Here is a simple back-tracking recursive algorithm for this problem, based on the two cases: If z is an interleaving, the first character in z is either copied from x or y.

```
BTInter($x_1, ..., x_n$ ; $y_1, ..., y_m$; $z_1, ..., z_{n+m}$):
  - IF $n = 0$ THEN IF $y_1, ..., y_m = z_1, ..., z_m$ THEN return True ELSE return False
  - IF $m = 0$ THEN IF $x_1, ..., x_n = z_1, ..., z_n$ THEN return True ELSE return False
  - IF $x_1 = z_1$ AND BTInter($x_2, ..., x_n$, $y_1, ..., y_m$; $z_2, ..., z_{n+m}$) return True
  - If $y_1 = z_1$. AND BTInter($x_1, ..., x_n$, $y_2, ..., y_m$; $z_2, ..., z_{n+m}$) return True
  - Return False
```

   Answer the parts below:

   (a) (3 points) Show the tree of recursions the above algorithm would make on the above example.

   (b) (3 points) Give an upper bound on the total number of recursive calls this algorithm might make in terms of n and m.

   (c) (3 points) Which distinct sub-problems can arise in the recursive calls for this problem?

   (d) (7 points) Translate the recursive algorithm into an equivalent DP algorithm, using your answer to part (c).

   (e) (4 points) Give a time analysis for your DP algorithm

NOTE: For the remaining questions, structure your answer in the following format. You should explicitly give:

1. Description of sub-problems (2 points)
2. Base Case(s) (2 points)
3. Recursion with justification. (*A complete proof by induction is NOT required. However, you should explain why the recursion makes sense and how it covers all possibilities*) (6 points)
4. Order in which sub-problems are solved (2 points)
5. Form of output (how do we get the final answer?) (2 points)
6. Pseudocode (3 points)
7. Runtime analysis (3 points)
8. A small example explained using an array or matrix as in the previous questions (Optional)

---

2. (20 points) A subsequence is *palindromic* if it is the same whether read left to right or right to left. For instance, the sequence

$$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

has many palindromic subsequences, including A, C, G, C, A and A, A, A, A (on the other hand, the subsequence A, C, T is *not* palindromic). Devise an algorithm that takes a sequence $x[1 \dots n]$ and returns the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

3. (20 points) You are going on a long trip. You start on the road at milepost 0. Along the way there are n hotels, at mileposts $a_1 < a_2 < \cdots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which hotels you stop at. You must stop at the final hotel (at a distance a_n), which is your destination.

   Ideally, you'd like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the *penalty* for that day is $(200 - x)^2$. You want to plan your trip to minimise the total penalty—that is, the sum, over all travel days, of the daily penalties.

   Give an efficient algorithm that determines the optimal sequence of hotels at which to stop.

4. (20 points) You are given an $n \times 4$ matrix A consisting of integers (positive or negative). Your goal is to find a set S of tuples (i, j) indicating locations of the 2-D matrix A such that:

   1. $\sum_{(i,j) \in S} A[i, j]$ is maximized, and
   2. For all pairs of tuples $(i_1, j_1), (i_2, j_2) \in S$, $(i_2, j_2) \notin \{(i_1 - 1, j_1), (i_1 + 1, j_1), (i_1, j_1 - 1), (i_1, j_1 + 1)\}$.

   (*For example, consider the* 2×4 *matrix below. The set of locations that satisfies (1) and (2) above are indicated by shading these locations in the matrix.*)

-1	2	**3**	-4
2	-5	3	**5**

   Design a DP algorithm for this problem that outputs the maximum possible sum attainable.

5. (20 points) A town has n residents labelled $1, ..., n$. All n residents have houses along a single road. The town authorities suspect a virus outbreak and would like to set up k testing centers along this road. They want to set up these k testing centers in locations that minimise the sum of squared distance that all the residents need to travel from their house to get to their nearest testing center. Moreover, the centers will be opened in residents' houses to reduce the operating cost. None of the residents has any objection if a center is opened in their house. You have been asked to design an algorithm for finding the optimal locations of the k testing centers.

   Since all residents live along a single road, the house location of a resident can be identified by the distance along the road from a single reference point (which can be thought of as the starting point of the town). As input, you are given integer n, integer k, and the house location of the residents in an integer array $A[1...n]$ where $A[i]$ denotes the house location of resident i. Moreover, $A[1] \leq A[2] \leq A[3] \leq ... \leq A[n]$. Your algorithm should output an integer array $C[1...k]$ of house numbers such that the following quantity gets minimised:

$$\sum_{i=1}^{n} D(i), \text{ where } D(i) = \min_{j \in \{1,\dots,k\}} (A[i] - A[C[j]])^2$$

Note that $D(i)$ denotes the squared distance resident i has to travel to get to the nearest testing center out of centers at houses $C[1], ..., C[k]$.

(*For example, consider* $k = 2$ *and* $A = [1, 2, 3, 7, 8, 9]$. *A solution for this case is* $C = [2, 5]$. *Note that for testing centers at locations* 2 *and* 8, *the total squared distance travelled by residents will be* $(1 + 0 + 1 + 1 + 0 + 1) = 4$.)

Design a DP algorithm for this problem that outputs the minimum achievable value of the squared distance.