The instructions are the same as in Homework-05.

There are 5 questions for a total of 100 points.

1. (20 points) (Hamiltonian path)
   Consider the following algorithm for deciding whether an undirected graph has a Hamiltonian Path from
   $x$ to $y$, i.e., a simple path in the graph from $x$ to $y$ going through all the nodes in $G$ exactly once. ($N(x)$
   is the set of neighbors of $x$, i.e. nodes directly connected to $x$ in $G$).

   1. $\text{HamPath}(G, x: \text{node}, y: \text{node})$
   2. If $x = y$ is the only node in $G$ return $True$.
   3. If no node in $G$ is connected to $x$, return $false$.
   4. For each $z \in N(x)$ do:
   5. If $\text{HamPath}(G - \{x\}, z, y)$, return $true$.
   6. Return $false$

   a. Explain (informally) why this algorithm is correct. (5 points)
   b. If every node of the graph $G$ has degree (number of neighbors) at most 3, how long will this algorithm
      take at most? (15 points) (Hint: you can get a tighter bound than the most obvious one.)

2. (20 points) Consider two binary strings $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$. An interleaving of $x$ and $y$ is a strings
   $z_1, \ldots, z_{n+m}$ 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 $y = 0011$, $z = 10001101$ 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, either the first character in $z$ is either copied from $x$ or from $y$.

   BTInterleaving $(x_1, \ldots, x_n; y_1, \ldots, y_m; z_1, \ldots, z_{n+m})$:
   1. IF $n=0$ THEN IF $y_1, \ldots, y_m = z_1, \ldots, z_m$ THEN return $True$ ELSE return $False$
   2. IF $m=0$ THEN IF $x_1, \ldots, x_n = z_1, \ldots, z_n$ THEN return $True$ ELSE return $False$
   3. IF $x_1 = z_1$ AND BTInterleaving $(x_2, \ldots, x_n, y_1, \ldots, y_m; z_2, \ldots, z_{n+m})$ return $True$
   4. IF $y_1 = z_1$ AND BTInterleaving $(x_1, \ldots, x_n, y_2, \ldots, y_m; z_2, \ldots, z_{n+m})$ return $True$
   5. Return $False$

   a. (2 points): Show the tree of recursions the above algorithm would make on the above example.
   b. (4 points). Give an upper bound on the total number of recursive calls this algorithm might make in
      terms of $n$ and $m$
   c. (2 points). Which distinct sub-problems can arise in the recursive calls for this problem?
   d. (6 points): Translate the recursive algorithm into an equivalent DP algorithm, using your answer to
      part c.
   e. (3 points) Give a time analysis for your DP algorithm
   f. (3 points) Give the matrix your DP algorithm would produce on the above example.
NOTE: For questions 3, 4 and 5, 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 point)
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)

3. (20 points) A subsequence is palindromic if it is the same whether read left to right or right to left. For instance, the sequence


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 \( x[1 \ldots n] \) and returns the (length of the) longest palindromic subsequence. Its running time should be \( O(n^2) \).

4. (20 points) You are going on a long trip. You start on the road at mile post 0. Along the way there are \( n \) hotels, at mile posts \( a_1 < a_2 < \cdots < a_n \), 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 of the hotels you stop at. You must stop at the final hotel (at distance \( a_n \)), which is your destination.

You'd ideally 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 so as to minimize 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.

5. (20 points) (K-shot strategy) Let \( p(1), \ldots, p(n) \) be prices of a stock for \( n \) consecutive days. A \( k \)-shot strategy is a collection of \( m \) pairs of days \( (b_1, s_1), \ldots, (b_m, s_m) \) with \( 0 \leq m \leq k \) and \( 1 \leq b_1 < s_1 < \cdots < b_m < s_m \leq n \). For each pair of days \( (b_i, s_i) \), the investor buys 100 shares of stock on day \( b_i \) for a price of \( p(b_i) \) and then sells them on day \( s_i \) for a price of \( p(s_i) \) with a total return of:

\[
100 \sum_{1 \leq i \leq m} p(s_i) - p(b_i)
\]

Design a Dynamic Programming algorithm that takes as input the prices of the \( n \) consecutive days, \( p(1), \ldots, p(n) \) and for some \( k \) computes the maximum return among all \( k \)-shot strategies.

(Hint: Your sub-problems should be indexed by two indices.)