- The instructions are the same as in Homework 1, 2, 3 and 4.

There are 5 questions for a total of 88 points.

---

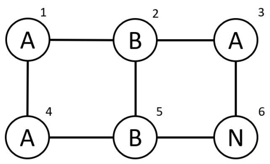
1. (*Abstract Word Search*) Given a directed graph G where each vertex has out-degree at most d, and each vertex v is labelled by a symbol $\ell(v)$, a start vertex v_1, and a sequence of symbols $w_1..w_k$, is there a path $v_1..v_k$ in G starting from v_1, so that $\ell(v_1)\ell(v_2)...\ell(v_k) = w_1...w_k$? *Note that the path might not be a simple path.*

   Here is a recursive back-tracking algorithm that solves *Abstract Word Search*:

```
BTAWS($G, v, w_1..w_k$)
    IF $\ell(v) \neq w_1$ return False
    IF $k = 1$ return True
    FOR $u \in N(v)$ do:
        IF BTAWS($G, u, w_2..w_k$)==True THEN return True
    Return False
```

   $N(v)$ in the above pseudocode denotes the neighboring vertices of vertex v (i.e., vertices to which v has an out-going edge). Answer the questions that follow:

   (a) (2 points) Illustrate the tree of recursive calls the above algorithm makes on the input (G, 2, "$BANANA$") (Note: The graph G given below, the starting vertex v is vertex 2, and $w_1..w_6 = BANANA$).



   (b) (2 points) Give an upper bound on the number of total recursive calls the above procedure makes on a general input.

   (c) (2 points) Characterize the different subproblems that arise in the BT algorithm above.

   (d) (2 points) Define an array or matrix based on the sub-problem characterization of part (c).

   (e) (4 points) Compose the base cases of your array and express your array entries recursively. (give a brief justification of why your recursion works.)

   (f) (2 points) Convert the above backtracking algorithm into a dynamic programming algorithm.

   (g) (2 points) Analyze the running time of 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*) (5 points)

4. Order in which sub-problems are solved (2 points)
5. Form of output (how do we get the final answer?) (1 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. (18 points) (*Laying rugs*) You have a long corridor in your palace. Unfortunately, there are n unsightly marks on the corridor floor, at $x_1, ..., x_n$ yards from the door. You will use some costly rugs whose width fills the corridor and are each one yard long to cover the marks. For simplicity, we will allow rugs to overlap and part of the rug to extend past the end of the corridor. We identify rugs with intervals of length 1, and it hides the mark at x_i if x_i is in the interval. Unfortunately, we have a fixed number of rugs R, and the goal is to use them to hide as many marks as possible. Design an algorithm to minimize the number of unhidden marks, given $x_1, ..., x_n$, and R as inputs. Your algorithm only needs to output the minimum number of unhidden marks possible.

3. (18 points) (*Palindromic subsequence*) 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 $x[1 \ldots n]$ and returns the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

4. (18 points) (*Shipping boxes*) You are a manager at a shipping company. On a particular day, you must ship n boxes with weights $w[1...n]$ that are positive integers between 1 and 100. You have three trucks, each with a weight limit of D, a positive integer. Design an algorithm to determine if it is possible to pack all the n boxes into the three trucks such that for every truck, the total weight of boxes in the truck is at most D (*i.e., the weight limit is not exceeded*). Your algorithm should output "yes" if it is possible to pack and "no" otherwise.

5. (18 points) (*Testing centers*) A town has n residents labeled $1, ..., n$. 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 minimize the sum of squared distance that all the residents need to travel from their house 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 town's starting point). 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 aim to find an integer array $C[1...k]$ of house numbers such that the following quantity gets minimized:

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

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.