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

There are 6 questions for a total of 100 points.

---

1. Consider two binary strings $x = x_1, ...x_n$ and $y = y_1, ..., y_m$. An interleaving of x and y is a string $z = z_1, ..., z_{n+m}$ 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 from 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) (2 points) Give an upper bound on the total number of recursive calls this algorithm might make in terms of n and m.

   (b) (2 points) Which distinct sub-problems can arise in the recursive calls for this problem?

   (c) (4 points) Translate the recursive algorithm into an equivalent DP algorithm, using your answer to part (c).

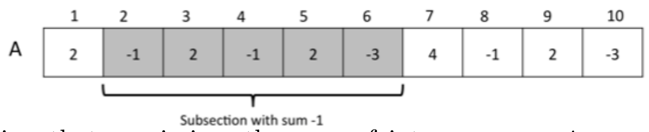
   (d) (2 points) Give a time analysis for your DP algorithm

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

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 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)

---

2. (18 points) Given a sequence of integers (positive or negative) in an array $A[1...n]$, the goal is to find a *subsection* of this array such that the sum of integers in the subsection is maximized. A subsection is a contiguous sequence of indices in the array. (*For example, consider the array and one of its subsections below. The sum of integers in this subsection is* -1.)



Let us call a subsection that maximizes the sum of integers a *maximum subsection.* Design a DP algorithm for this problem that output the sum of numbers in a maximum subsection.

3. (18 points) Let $p(1), \dots, p(n)$ be prices of a stock for n consecutive days. A k-block strategy is a collection of m pairs of days $(b_1, s_1), \dots, (b_m, s_m)$ with $0 \leq m \leq k$ and $1 \leq b_1 < s_1 < \dots < b_m < s_m \leq n$. 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 DP algorithm that takes as input a positive integer k and the prices of the n consecutive days, $p(1), \dots, p(n)$ and computes the maximum return among all k-block strategies.

4. (18 points) You are given an $n \times 5$ 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* 2×5 *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	▨ 5
▨ 2	-5	3	▨ 5	6

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

5. (18 points) A town has n residents labelled $1, ..., n$. All n residents live along a single road. The town authorities suspect a virus outbreak and want to set up k testing centers along this road. They want to set up these k testing centers in locations that minimises the total sum of distance that all the residents need to travel to get to their nearest testing center. 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 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 location of the residents in an integer array $A[1...n]$ where $A[i]$ denotes the 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 locations such that the following quantity gets minimised:

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

Here $|x - y|$ denotes the absolute value of the difference of numbers x and y. Note that $D(i)$ denotes the distance resident i has to travel to get to the nearest testing center out of centers at $C[1], ..., C[k]$.

(*For example, consider* $k = 2$ *and* $A = [1, 2, 3, 7, 8, 9]$. *A solution for this case is* $C = [2, 8]$. *Note that for testing centers at locations* 2 *and* 8, *the total 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 total distance.

6. (18 points) 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, which is a positive integer. Design an algorithm to determine if it is possible to pack all the n boxes into the three trucks such that the 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. The running time of your algorithm should be polynomial in n and D.