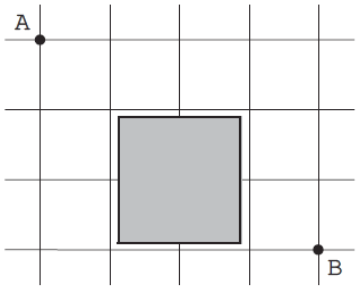
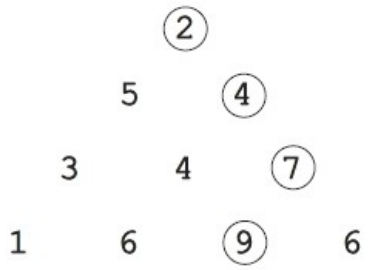
Solve the following assigned problems by both Dynamic Programming and Greedy Paradigms. Justify your solution.

[22551] Super Egg Problem: A firm has invented a super-strong egg. For publicity purposes, it wants to determine the highest floor in a 100-story building from which such an egg can fall without breaking. The firm has given a tester two identical eggs to experiment with. Of course, the same egg can be dropped multiple times until it breaks.

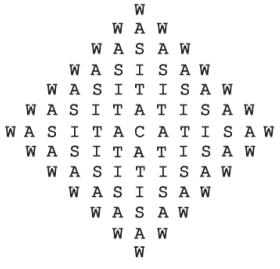
[22552] Blocked Paths: Find the number of different shortest paths from point A to point B in a city with perfectly horizontal streets and vertical avenues as shown in Figure given below. No path can cross the fenced off area shown in grey in the figure.



[22553] Maximum Sum Descent: Some positive integers are arranged in a triangle like the one shown in the Figure given below. Design an algorithm (more efficient than an exhaustive search of course) to find the largest sum in a descent from its apex to the base through a sequence of adjacent numbers, one number per each level.



[22554] Palindrome Counting: In how many different ways can the palindrome: **WAS IT A CAT I SAW**, be read in the diamond-shaped arrangement shown in the Figure given below? You may start at any W and



go in any direction on each step - up, down, left, or right - through adjacent letters. The same letter can be used more than once in the same sequence.

[22555] Breaking a string: A certain string-processing language allows a programmer to break a string into two pieces. Because this operation copies the string, it costs n time units to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks occur can affect the total amount of time used. For example, suppose that the programmer wants to break a 20-character string after characters 2, 8, and 10 (numbering the characters in ascending order from the left-hand end, starting from 1). If she programs the breaks to occur in left-to-right order, then the first break costs 20 time units, the second break costs 18 time units (breaking the string from characters 3 to 20 at character 8), and the third break costs 12 time units, totaling 50 time units. If she programs the breaks to occur in right-to-left order, however, then the first break costs 20 time units, the second break costs 10 time units, and the third break costs 8 time units, totaling 38 time units. In yet another order, she could break first at 8 (costing 20), then break the left piece at 2 (costing 8), and finally the right piece at 10 (costing 12), for a total cost of 40. Design an algorithm that, given the numbers of characters after which to break, determines a least-cost way to sequence those breaks. More formally, given a string S with n characters and an array $L[1..m]$containing the break points, compute the lowest cost for a sequence of breaks, along with a sequence of breaks that achieves this cost.

[22556] Pile Splitting: Given n counters in a pile, split the counters into two smaller piles and compute the product of the numbers of the counters in the two piles obtained. Continue to split each pile into two smaller piles and to compute the products until there are n piles of size one. Once there are n piles, sum all the products computed. How should one split the piles to maximize the sum of the products? What is the maximal sum equal to?

[22557] Jack Straws: In the game of Jack Straws, a number of plastic or wooden "straws" are dumped on the table and players try to remove them one-by-one without disturbing the other straws. Here, we are only concerned with whether various pairs of straws are connected by a path of touching straws. Given a list of the end points for n>1 straws (as if they were dumped on a large piece of graph paper), determine all the pairs of straws that are connected. Note that touching is connecting, but also that two straws can be connected indirectly vie other connected straws.

[22558] Box Stacking: You are given a set of n types of rectangular 3-D boxes, where the i^{th} box has heighth_i, width w_i and depth d_i (all real numbers). You want to create a stack of boxes which is as tall as possible, but you can

only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is disallowed to use multiple instances of the same type of box.

[22559] Printing Neatly: Consider the problem of neatly printing a paragraph on a printer. The input text is a sequence of n words of lengths $l_1, l_2, \ldots, l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows: if a given line contains words i through j and we leave exactly one space between words, the number of extra space characters at the end of the line is the difference between M and the total number of characters in the words plus the spaces between them. We wish to print a paragraph of n words neatly on a printer minimizing the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines.

[22560] Picking Up Coins: Some coins are spread in the cells of an $n \times m$ board, one coin per cell. A robot, located in the upper left cell of the board, need to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it picks up that coin. Devise an algorithm to find the maximum number of coins the robot can collect and a path it needs to follow to do this.

[22561] Optimal Strategy for a Game: Consider a row of n coins of values $v_1 \ldots v_n$, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first. Also, print the sequence of moves in the optimal game. As many sequences of moves may lead to the optimal answer, you may print any valid sequence.

[22562] Planning an investment strategy: Your knowledge of algorithms helps you obtain an exciting job with the Acme Computer Company, along with a $10,000 signing bonus. You decide to invest this money with the goal of maximizing your return at the end of 10 years. You decide to use the Amalgamated Investment Company to manage your investments. Amalgamated Investments requires you to observe the following rules. It offers n different investments, numbered 1 through$n$. In each year j, investment$i$provides a return rate of r_{ij} . In other words, if you invest d dollars in investment i in year j , then at the end of year j, you have $d\,r_{ij}$ dollars. The return rates are guaranteed, that is, you are given all the return rates for the

next 10 years for each investment. You make investment decisions only once per year. At the end of each year, you can leave the money made in the previous year in the same investments, or you can shift money to other investments, by either shifting money between existing investments or moving money to a new investment. If you do not move your money between two consecutive years, you pay a fee of f_1 dollars, whereas if you switch your money, you pay a fee of f_2 dollars, where $f_2 > f_1$. Design an algorithm that plans your optimal investment strategy.

[22563] Inventory planning: The Rinky Dink Company makes machines that resurface ice rinks. The demand for such products varies from month to month, and so the company needs to develop a strategy to plan its manufacturing given the fluctuating, but predictable, demand. The company wishes to design a plan for the next n months. For each month i , the company knows the demand d_i, that is, the number of machines that it will sell. Let $D = \sum_{i=1}^{n} d_i$ be the total demand over the next n months. The company keeps a full-time staff who provide labor to manufacture up to m machines per month. If the company needs to make more than m machines in a given month, it can hire additional, part-time labor, at a cost that works out to c dollars per machine. Furthermore, if, at the end of a month, the company is holding any unsold machines, it must pay inventory costs. The cost for holding j machines is given as a function $h(j)$ for $j = 1, 2, \cdots, D$, where $h(j) \geq 0$ for $1 \leq j \leq D$ and $h(j) \leq h(j+1)$ for $1 \leq j \leq D-1$. Give an algorithm that calculates a plan for the company that minimizes its costs while fulfilling all the demand.

[22564] Crossing Puzzle: A group consisting of n cannibals and n missionaries seeks to cross a river. A boat is available which will hold at most 3 people, and which can be navigated by any combination of cannibals and missionaries involving one or two people. If the missionaries on either side of the river, or in the boat, are outnumbered at any time by cannibals, the cannibals will indulge in their anthropophagic tendencies and do away with the missionaries. What minimum schedule of crossings can be devised to permit the entire group of cannibals and missionaries to cross the river safely?

[22565] Image compression by Seam Carving: We are given a color picture consisting of an $m \times n$ array $A[1..m, 1..n]$ of pixels, where each pixel specifies a triple of red, green, and blue (RGB) intensities. Suppose that we wish to compress this picture slightly. Specifically, we wish to remove one pixel from each of the m rows, so that the whole picture becomes one pixel narrower. To avoid disturbing visual effects, however, we require that the pixels removed in two adjacent rows be in the same or adjacent columns; the

pixels removed form a “seam” from the top row to the bottom row where successive pixels in the seam are adjacent vertically or diagonally.

Suppose now that along with each pixel $A[i, j]$, we have calculated a real-valued disruption measure $d[i, j]$, indicating how disruptive it would be to remove pixel $A[i, j]$. Intuitively, the lower a pixel’s disruption measure, the more similar the pixel is to its neighbors. Suppose further that we define the disruption measure of a seam to be the sum of the disruption measures of its pixels. Give an algorithm to find a seam with the lowest disruption measure.