Lab 5: Dynamic Programming

Q1. The problem of maximum sum of a path in a right number triangle.

Problem statement: Given a right triangle of numbers, apply the Dynamic programming approach to find the largest sum of numbers that appear on a path starting from the top towards the base. The next number on the path is located directly below or below-and-one-place-to-the-right.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the length of the triangle's vertical leg.
- Each of the n next lines show the integers that form the triangle. Two consecutive numbers are separated by a single space " "

Output: Print the output data to the console as follows

- The numbers on the specified path
- The corresponding largest sum of numbers.

Example of input and output:

Input from user keyboard	Output to console
7 8 8 4 6 0 7 0 8 5 0 5 4 9 4	7 8 6 8 9 38

Q2. The change-making problem.

Problem statement: Given k denominations: $d_1 < d_2 <..< d_k$ where $d_1 = 1$. Apply the Dynamic programming approach to find the minimum number of coins (of certain denominations) that add up to a given amount of money n.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer k to indicate the number of denominations.
- 2nd line: k positive integers describing k denominations, sorted descending, two consecutive numbers are separated by a single space " ". The last value must be one.
- 3rd line: a positive integer n to indicate the amount of money required exchange.

Output: If there exists a solution, print the amount of each denomination to the console. Otherwise, output the string "No solution".

Example of input and output:

Input text file	Output to console
4 25 10 5 1 72	25: 2 10: 2 5: 0 1: 2

Q3. The Longest monotonically increasing subsequence (LMIS) problem.

Problem statement: Apply the Dynamic programming approach to find the longest subsequence of a given sequence of positive integers such that all elements of the subsequence are in monotonically increasing order.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the size of the input array
- 2nd line: n integers, two consecutive numbers are separated by a single space " ".

Output: Print to the console the numbers in the longest monotonically increasing subsequence.

Example of input and output:

Input text file	Output to console
4 8 9 12 10 11	8 9 10 11

Q4. The Longest common subsequence (LCS) problem.

Problem statement: Apply the Dynamic programming approach to find the longest common subsequence of two given strings, $S = s_1 s_2 \dots s_m$ and $T = t_1 t_2 \dots t_m$.

Input: The user inputs the two strings, S and T.

Output: Print to the console their longest common subsequence.

Example of input and output:

Input from user keyboard	Output to console
XYXZPQ YXQYXP	XYXP

Q5. The Floyd's algorithm for the all-pairs shortest-paths problem.

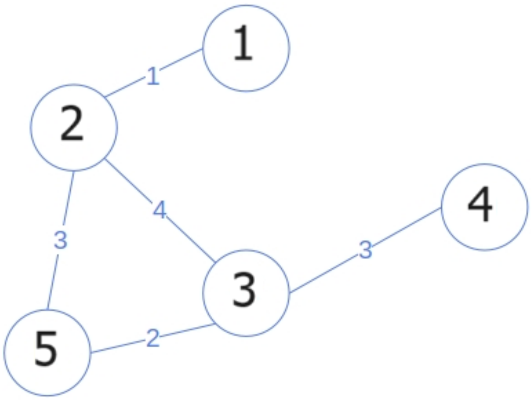
Problem statement: Given a weighted connected graph (undirected or directed) graph $G = (V, E)$. Apply the Dynamic programming approach to find the distances, i.e., the lengths of the shortest paths from each vertex to all other vertices.

Input: Read the input data, which is an adjacency list of the graph $G = (V, E)$, from a text file whose format is as follows

- 1st line: a positive integer n to indicate the number of vertices in the set of vertices V
- Each of the following lines presents the list of vertices arrived from a vertex, whose index ranges from 1 to n.
  - A non-negative integer m_i to indicate the number of vertices arrived from vertex i.
  - $2 \times m_i$ subsequent non-negative integers are indices and weights of the vertices arrived from vertex i. Each pair of consecutive numbers is for a vertex.
  - Please mind the direction of the edges. Two consecutive numbers are separated by a space.

Output: Print to the console all the shortest paths and their corresponding lengths.

Example of input and output:

<table>
<tr><th>Visualization</th><th>Input text file</th><th>Output to console</th></tr>
<tr><td>
</td><td>5
1 2 1
3 1 1 3 4 5 3
3 2 4 4 3 5 2
1 3 3
2 2 3 3 2</td><td>1 – 2: 1
1 – 2 – 3: 5
1 – 2 – 3 – 4: 8
1 – 2 – 5: 4
2 – 1: 1
2 – 3: 4
2 – 3 – 4: 7
2 – 5: 3
3 – 2 – 1: 5
3 – 2: 4
3 – 4: 3
3 – 5: 2
4 – 3 – 2 – 1: 8
4 – 3 – 2: 7
4 – 3: 3
4 – 3 – 5: 5
5 – 2 – 1: 3
5 – 2: 3
5 – 3: 2
5 – 3 – 4: 5</td></tr>
</table>

Q6. The Matrix chain multiplication problem.

Problem statement: Apply the Dynamic programming approach to find the most efficient way to multiply a sequence of n matrices $A_1 \times A_2 \times \ldots \times A_n$. Assume that the sizes (in order) of these matrices are $d_0 \times d_1, d_1 \times d_2, \ldots, d_{n-1} \times d_n$, respectively.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the number of matrices.
- Each of the n next lines show the two integers that represent the dimensions of a matrix. Two consecutive numbers are separated by a single space " ".

Output: Print to the console the most efficient way to multiply the sequence of matrices and the corresponding number of operations.

Example of input and output:

Input text file	Output to console
4 50 20 20 1 1 10 1 100	(A1 A2) (A3 A4) 7000

Q7. The optimal binary search trees.

Problem statement: Let $k_1, k_2, \ldots, k_n$ be keys of a binary search tree ($k_1 < k_2 < \cdots < k_n$) and $p_1, p_2, \ldots, p_n$ be the probabilities of searching for elements. Apply the Dynamic programming approach to find an optimal binary search tree for which the average number of comparisons in a search is the smallest possible value.

Note: For simplicity, we limit our discussion to minimizing the average number of comparisons in a successful search.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the number of keys (or probabilities).
- 2nd line: n consecutive integers sorted in increasing order, each of which is a key of the binary search tree.
- 3rd line: n consecutive real numbers, each of which is a probability of searching for elements.
- Two consecutive numbers are separated by a single space " ".

Output: Print to the console the pairs of ⟨key, probability⟩ and the total cost.

Example of input and output:

Input text file	Output to console
3 1 2 3 0.7 0.3 0.1	1 0.7 2 0.3 3 0.1 1.4

Q8. The sum of subsets problem.

Problem statement: Given a set of n distinct positive integers, $A = \{a_1, a_2, .., a_n\}$. Apply the Dynamic programming approach to find a subset of A that sum to a positive integer k.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the size of the input array
- 2nd line: n integers, two consecutive numbers are separated by a single space " "
- 3rd line: a positive integer k

Output: If there exists a solution, print all the subsets to the console, one per line, and two consecutive numbers in a subset are separated by a single space " ". Otherwise, print the string "No solution".

Example of input and output:

Input text file	Output to console
4 3 5 6 7 15	3 5 7

Q9. The Knapsack problem.

Problem statement: Given n items of known weights, $\{w_1, w_2, .., w_n\}$, and their corresponding values, $\{v_1, v_2, .., v_n\}$, and a knapsack of capacity C. Apply the Dynamic programming approach to find the most valuable subset of the items that fit into the knapsack.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer C to indicate the capacity of the knapsack.
- 2nd line: a positive integer n to depict the number of items.
- Each of n following lines represent the weight w_i and value v_i of the item i, where $i = 1, .., n$. The two values are separated by a single line " ".

Output: If there exists a solution, print the output data to the console as follows. Otherwise, print the string "No solution".

- The list of chosen items, whose sum of weights equals C
- The total value of the chosen items

Example of input and output:

Input text file	Output to console
20 5 10 5 4 2 9 4 6 6 7 1	1 2 4 13

Q10. The traveling salesman problem.

Problem statement: Given a set of n cities, some pairs of adjacent cities are connected with given distances. Apply the Dynamic programming approach to find the shortest tour through n cities such that we visit each city exactly once before returning to the city where we started.

Input: Read the input data from a text file whose format is as follows

- 1st line: a positive integer n to indicate the number of cities. The cities are indexed from 1 to n.
- Each of the next lines shows a pair of cities and their distance, $\langle point1\ point\ 2\ distance \rangle$, two consecutive numbers are separated by single space " ".
- The last line shows the number -1, indicating the end of the file

Output: If there exists a solution, print the below output data to the console. Otherwise, print the string "No solution".

- A sequence of cities forming the tour (in their exact order)
- The length of the tour

Example of input and output:

Input text file	Output to console
5 1 2 4 1 4 8 3 4 2 2 5 1 3 5 1 -1	1 2 5 3 4 16

Regulations for completing the lab work

- Each question must be implemented as an independent program in a single C++ file (of format .cpp).
- The program must receive input and return output as specified Submissions with wrong regulation will result in a "0" (zero).
- Plagiarism and Cheating will result in a "0" (zero) for the entire course.
- Contact: [Here]().