Lab #1: Brute-Force

Students implemented the following problems using "Brute-Force" technique.

1. Implemented **Shaker Sort** ascending on real number array.

Input	Output
Content of the *"input_ 1.txt"* file: - 1^{st} line: positive integer `n` - size of the input array. - 2^{nd} line: `n` real numbers, separated by single space " ".	Sorted array.
Example: 5 3.3 4.4 1.1 5.5 2.2	1.1 2.2 3.3 4.4 5.5

2. Find the subsequence with largest sum of elements in an given real array.

   Requirement: Implement using Dynamic programming version.

Input	Output
Content of the *"input_ 2.txt"* file: - 1^{st} line: positive integer `n` - size of the input array. - 2^{nd} line: `n` real numbers, separated by single space " ".	- 1^{st} line: Subsequence with largest sum of elements. - 2^{nd} line: Sum of the subsequence.
Example: 6 -1.3 2.1 4.4 -2 5.3 -1.1	2.1 4.4 -2 5.3 9.8

3. **The change-making problem**: Given `k` denominations“ $x_1, x_2, ..., x_k$. Find the minimum number of coins (of certain denominations) that add up to a given amount of money `n`. *Note*: Always assume that the smallest coin denomination is 1.

Input	Output
Content of the *"input_ 3.txt"* file: - 1^{st} line: `k` positive integers represent `k` denominations, sorted descending, separated by single space " ". The last value must be `1`. - 2^{nd} line: Positive integer `n` represents the amount of money required exchange.	- *Denominations 1: amount* - *Denominations 2: amount* - *...* - *Denominations k: amount*
Example: 25 10 5 1 72	25: 2 10: 2 5: 0 1: 2

4. **Convex-hull problem**: Given a set of n points, find the smallest convex polygon that contains all of them either inside or on its boundary.

   Requirement: Using the `computeAngle` Brute-Force approach.

Input	Output
Content of the *"input_4.txt"* file: - 1st line: positive integer `n` - amount of points - `n` next lines: Each line represents coordinates of a point, separated by single space " ".	Set of points belong to the convex-hull
Example: 7 0 0 1 1 3 3 1 3 0 4 4 0 4 4	(0 , 0) (4 , 0) (4 , 4) (0 , 4)

5. **Traveling Salesman Problem**: The problem asks to find the shortest tour through a given set of `n` cities that visits each city exactly once before returning to the city where it started.

Input	Output
Content of the *"input_5.txt"* file: - 1st line: positive integer `n` represents `n` cities. Cities are numbered from 1 → n - Following lines: City 1 City 2 Distance - Last line: -1	- 1st line: Traveling order (shortest path). - 2nd line: Length of traveled path.
Example: 5 1 2 4 1 4 8 3 4 2 2 5 1 3 5 1 -1	1 2 5 3 4 17

6. **Knapsach problem**: Given n items of known weights $w_1, w_2, ..., w_n$ and values $v_1, v_2, ..., v_n$ and and a knapsack of capacity W. Find the most valuable subset of the items that fit into the knapsack.

Input	Output
Content of the *"input_ 6.txt"* file: - 1st line: positive integer W represents capacity of the knapsack. - 2nd line: positive integer n represents items. These items are numbered from 1 → n - n following lines: w_i v_i (represent weight and value of item i)	- 1st line: ID of the chosen items. - 2nd line: Total value of th chosen items
Example: 20 5 10 5 4 2 9 4 6 6 7 1	1 2 4 13

- **FILE SUBMISSION REGULATION**
  - Only submit files with .cpp extensions: 1.cpp, 2.cpp, ... **Project submission is illegal**.
  - .cpp files must be located in MSSV folder, then be compressed into MSSV.zip(.rar).
  - Source code must receive input and return output as specified for each problem. Submissions with wrong regulation will result in a "0" (zero).
  - Plagiarism and Cheating will result in a "0" (zero) for the entire course.
  - Contact: **bhthong@fit.hcmus.edu.vn** for more information.

HẾT