CS 372/469 – Spring 2022
Lab 5
Due: 04/30/2022 11:59 pm

For each of the following questions, write a successful running code in any programming language that you prefer. Your code should run without any errors for any *valid* input.

All problems are borrowed from GeeksForGeeks.

## Question 1 (30 points):

The goal of this question is to implement the 0-1 Knapsack problem without repetition using Dynamic Programming **Tabulation (Bottom-up) method**.

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Source (Open at your own risk – solution is also on this page): https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

Input and Output Example (from the above source):
Input will include Items as (value, weight) pairs and Knapsack capacity W
*{{60, 10}, {100, 20}, {120, 30}}, W = 50*

*Output:*

```
Maximum possible value = 240
```

*Items to take: Item 1, 2*

Your code should be able to traverse the above input format (e.g. *{{60, 10}, {100, 20}, {120, 30}}, W = 50*) **from a given text file**.
Your algorithm must have a time complexity of O(N*W).
where 'N' is the number of elements and 'W' is capacity. As for every element, we traverse through all weight capacities 1<=w<=W.

## Question 2 (35 points):

The goal of this question is to implement the 0-1 Knapsack problem **with** repetition using Dynamic Programming **Tabulation (Bottom-up) method**.

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Source (Open at your own risk – solution is also on this page): https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

Input and Output Example (from the above source):
Input will include Items as (value, weight) pairs and Knapsack capacity W
*{{60, 10}, {100, 20}, {120, 30}}, W = 50*

*Output:*

```
Maximum possible value = 240
```

*Items to take: Item 1, 2*

Your code should be able to traverse the above input format (e.g. *{{60, 10}, {100, 20}, {120, 30}}, W = 50*) **from a given text file**.
Your algorithm must have a time complexity of O(N*W).
where 'N' is the number of elements and 'W' is capacity. As for every element, we traverse through all weight capacities 1<=w<=W.

## Question 3 (35 points):

The goal of this question is to find the longest common subsequence from the given two strings (explained from Slide 85 in Chapter 6 slides and https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/). You should use the Tabulation method.

Input: ABCBDAB, BDCABAB
Your code should be able to traverse the above input format **from a given text file**.

Your algorithm must have a time complexity of O(mn)

**Submission Instructions**: Put all your solutions in a properly commented file named *lab5_lastname_firstname.EXTENSION*, where EXTENSION = the appropriate extension for the programming language that you chose.