Assignment 4:

Due Date: Friday March 4th, 11:59 p.m.

Objectives:

- Greedy Algorithms
- Dynamic Programming
- Coding an algorithm

Submission Instruction

- **Start early**
- You are allowed to work in groups of at most two students. It is okay if you want to work on your own.
- Write your answers to questions 1 to 5 in a file named **A4.pdf**
- Write your answers to programming questions in files named **P1.py** and **P2.py**
- Put the above three files in a folder named: **username_A4**, zip the folder and submit the zipped folder.
- Make sure to have one submission per group
- **[10 points]** Make sure that you are typing or your writing is neat and you follow submission instruction
- **For the programming questions, make sure your input and output format matches exactly with the examples given below**
- **Have your name(s) at the top of the pdf file and when you are submitting in MyLearningSpace**

Problems

[5 points] Q1. In the topic of Greedy Algorithms, we proved the optimality of the greedy algorithm for the coin change problem when US. coin denomination was used. Why does that proof not work for the U.S. post office denomination? Remember: The greedy algorithm did not result in the optimal solution when the set of denominations was U.S. postage.

[5 points] Q2. In the topic of greedy algorithms, we solved the following problem: **Scheduling to minimize lateness.** Prove that this problem has the optimal substructure property.

Note: We talked about proving optimal substructure properties when talking about dynamic programming. You can use the technique discussed in dynamic programming here.

[5 points] Q3. Can we solve the above problem (scheduling to minimize lateness) using dynamic programming. If yes, how? If not, why?

[bonus: 10 points] **Q4.** Write the pseudocode for the first **programming question** described below and prove the optimality of the problem. You need to develop the pseudocode, although the most part of the points go to proving the optimality of your solution.

[15 points] Q5. Design and develop the pseudo code for the problem described in the second programming question below. Write the pseudo code using memoized technique, bottom-up technique and Analyze the time complexity of your algorithm in both bottom-up and memoized version.

Programming Questions

- **Your program must run correctly or you do not get much mark for it.**
- For the first one use greedy technique and for the second one you need to use dynamic programming technique. If you implement using the brute-force technique you do not get much mark. The program will be tested with large input values.

[10 points] P1. You are given N integers, and N-1 signs. Each of the signs is either a plus ('+') or a minus ('-'). You need to place the numbers in a sequence and then put a single sign between each pair of integers. The goal is to get the maximum value when we evaluate the expression. You can put the signs and the numbers in any order.

Here are a few examples. The first row of the input file consists of N integers, the second row specifies the number of plus signs and the third row specifies the number of minus signs. The sum of the number of plus signs and the number of minux signs is equal to N-1.

Example:

```
input.txt
-6,11,-13,17,5
1
3

output:
42
11 - (-6) - (-13) + 17 - 5 = 42
-(-6) +11 - (-13) +17 - 5 = 42
```

Write a program to take the name of the input file as an argument and print the output to the screen. Here is how we run your program:

```
>>python P1.py input.txt
```

Hint: this is a greedy algorithm and pretty easy to code up.

[10 points] P2. A product has packaging of different sizes. As the input you are given different sizes of the packaging of the product. You want to buy this product so that the total size is equal to a certain given value, **m**.
Develop an algorithm to return the minimum number of packages you must buy in order to purchase exactly **m** units. If it is not possible to do so, return -1.

Examples:

```
input.txt
2,3
6
```

```
output:
2
2 packages of size 3
```

```
input.txt
6,9,20
8
```

```
output:
-1
There’s no solution
```

```
input.txt
1,5,12
15
```

```
output:
3
3 packages of size 5
```

The program should read the input from a file. The file should be passed as an argument to the program. The output should be written on the screen. Your input and output should follow exactly the format given below:

```
>>python P1.py input.txt
```

Grading

[Total points: 60]

[10 points] Following submission instruction

[5 points] Q1

[5 points] Q2

[5 points] Q3

[bonus: 10 points] Q4

[15 points] Q5

[10 points] P1

[10 points] P2