`

CS 310 'Algorithms' **Assignment-4**

Due date: Monday, December 3, 2018 at 11:55 PM. [Total Marks = 70]

IMPORTANT: Read the instructions at the end of this document and follow the naming conventions.

DYNAMIC PROGRAMMING

Q1. **[10 marks]**

A factory produces solar panels for large businesses. The volume of panels produced depends on the orders placed by its customers. The machines at the factory can be run at low production capacity or high production capacity. If the machines are to be run at high capacity in some week (say week X), then they must be stopped and specially primed in the week X-1 and can't produce any panels during that time. The machines, however, can be run at low capacity for weeks without stopping. A high capacity week can also be immediately followed by a low capacity week. A high capacity job can be selected in Week 1 as it is assumed that the machines are already primed.

The factory manager maintains a record of high and low capacity orders for each week and must decide which one to pick each week or stop for priming. The goal is to select a combination of low and high orders so that the overall revenue for the factory is maximized. Let R_L denote the revenue for a low capacity job and R_H denote the revenue for a high capacity job.

Example: Suppose the factory manager maintains the following table. The revenue for week-1 would be 22 if the manager picks a high capacity order and 20 if a low capacity order is selected. Similarly for other weeks. Assume that the factory manager has this information for 'n' weeks, you have to find which combination of R_H and R_L will maximize the total revenue for n weeks.

	Week 1	Week 2	Week 3	Week 4	Week 5
R_H	22	4	100	200	100
R_L	20	15	65	100	60

A) Write an efficient dynamic-programming algorithm in C/C++ that determines what should be done in each of the n weeks (low job, high job or priming) to maximize the overall revenue. Print the selection for each week and the value of the revenue at the end of nth week. Clearly write the **recurrence relation** of the problem in the comments of your code. **You must clearly specify what your recurrence relation defines and what are the parameters that it takes.** [9]

B) What is the time complexity of your algorithm? Describe how many subproblems you've computed and how much time it takes to compute each subproblem. [1]

Your program will read the input from a text file. The first line of the file contains one positive integer value for n. The second line has n positive numbers for R_H and the third line has n positive numbers for R_L. See input format below.

Input format:

```
n 5
RH 22 4 100 200 100
RL 20 15 65 100 60
```

Output format:

```
Week 1: High 22
Week 2: Low 15
Week 3: Priming
Week 4: High 200
Week 5: Low 60
Total Revenue: 297
```

Q2. **[15 marks]**

You are given a ruler of length ‘n’, which has markings at unit length distances labeled 1 to n from left to right. This ruler is to be cut at ‘m’ positions. You are given the ‘m’ cut positions/labels on the ruler. Each cut has a different cost which depends on the order in which the cuts are made, as described in the following example.

Example:

> Suppose you are given a ruler of length n=15, which is to be cut at m=3 positions. The cuts should be made at labels {3, 5, 10}. *The cost of a cut is equal to the length of the ruler segment on which the cut is applied*.

- One way of cutting the ruler is that you decide to make the first cut at label 5. The ruler segment on which this first cut is made is the whole ruler, so the cost of the first cut is its length, i.e., 15. Now the ruler is in two pieces of lengths 5 and 10, let’s call them piece 1 and piece 2, respectively. You decide to make the second cut at label 3, which lies on piece 1. The length of piece 1 is 5 so cost of this second cut is 5. The third and last cut is made at label 10 which lies on piece 2 and its cost is equal to length of piece 2 i.e., 10. Total cost of cutting the ruler in this order is 15+5+10 = 30.
- Another way of cutting the ruler is that you decide to make the first cut at label 3. The ruler segment on which this first cut is made is the whole ruler, so the cost of the first cut is its length, i.e., 15. Now the ruler is in two pieces of lengths 3 and 12, let’s call them piece 1 and piece 2, respectively. You decide to make the second cut at label 5 which lies on piece 2. The length of piece 2 is 12 so cost of this second cut is 12. Now piece 2 is further divided into two pieces, let’s call them piece 2A of length 2 and piece 2B of length 10. The third and last cut is made at label 10 on piece 2B and its cost is equal to length of piece 2B i.e., 10. Total cost of cutting the ruler in this order is 15+12+10 = 37.
- Yet another way of cutting the ruler is that you decide to make the first cut at label 10. The ruler segment on which this first cut is made is the whole ruler, so the cost of the first cut is its length, i.e., 15. Now the ruler is in two pieces of lengths 10 and 5, let’s

`

call them piece 1 and piece 2, respectively. You decide to make the second cut at label 3 which lies on piece 1. The length of piece 1 is 10 so cost of this second cut is 10. Now piece 1 is further divided into two pieces, let's call them piece 1A of length 3 and piece 1B of length 7. The third and last cut is made at label 5 on piece 1B and its cost is equal to length of piece 1B i.e., 7. Total cost of cutting the ruler in this order is 15+10+7 = 32.

As you can see, there are other possible orderings as well. You have to find an **ordering** of cuts that has the **least cost**.

A) Write an efficient dynamic-programming algorithm in C/C++ that solves the above problem. The input to your program is the value of 'n' and 'm' cut positions, where m<n. You have to specify the optimal ordering of the cuts and its cost. Clearly write the **recurrence relation** of the problem in the comments of your code. **You must clearly specify what your recurrence relation defines and what are the parameters that it takes.** [14]

B) What is the time complexity of your algorithm? Describe how many subproblems you've computed and how much time it takes to compute each subproblem. [1]

You will read the input from a text file that contains the value of 'n' and the 'm' labels where the ruler is to be cut (see input format below).

Input Format:
n 15
m 3 5 15

Output Format:
Optimal cut ordering: 5 3 15
Least cost: 30
Note: More than one optimal cost orderings are possible.

Q3. **[15 marks]**

You are given a set of 'n' positive integers and an integer k. You have to determine if its possible to divide all the 'n' numbers into two disjoint sets S_1 and S_2, such that:

$\sum_{x} x \in S_1$ - $\sum_{y} y \in S_2$ = k (i.e. the difference of the sum of numbers in S_1 and the sum of numbers in S_2 is equal to k).

Note that S_1 and S_2 are disjoint sets and each of the 'n' numbers must either be in S_1 or in S_2 i.e., the $| S_1 \cup S_2 | = n$.

A) Write an efficient dynamic-programming algorithm in C/C++ that solves the above problem. The input to your program is the value of 'k' and 'n' positive integers. Your code should output "POSSIBLE" or "NOT POSSIBLE" based on the above criteria. If POSSIBLE, then print the set of numbers in S_1 and S_2 and their respective sums. Clearly write the **recurrence relation** of the problem in the comments of your code.

`

You must clearly specify what your recurrence relation defines and what are the parameters that it takes. [14]

B) What is the time complexity of your algorithm? Describe how many subproblems you've computed and how much time it takes to compute each subproblem. [1]

Input Format:

```
k 13
n 1 7 3 9 5 4
```

Output Format:

```
POSSIBLE
S1: 1 7 9 4   sum = 21
S2: 3 5   sum = 8
Difference: 21-8 = 13
```

Q4. [15 marks]

Let S1, S2 and S3 be three strings. String S1 is 'n' characters long, S2 is 'm' characters long and S3 is n+m characters in length. You have to find if S3 can be formed by some interleaving of all the characters in S1 and S2 such that the left to right ordering of the characters in each string is maintained.

Example: Let S1 = "merges" and S2 = "mired"

S3 = "mmiergreeds" is a valid interleaving.

S3 = "msergemired" is an invalid interleaving, because left to right ordering is violated and similarly S3 = "mirgesmered" is an invalid interleaving.

A) Write an efficient dynamic-programming algorithm in C/C++ that determines whether S3 is a valid interleaving of S1 and S2. Your code should output VALID or INVALID. If VALID, then mention the interleaving order (see output below). Clearly write the **recurrence relation** of the problem in the comments of your code. **You must clearly specify what your recurrence relation defines and what are the parameters that it takes.** [14]

B) What is the time complexity of your algorithm? Describe how many subproblems you've computed and how much time it takes to compute each subproblem. [1]

Your program will read the three strings from an input text file. String S1 will be on the first line of the file, S2 on the second and S3 on the third line. See input format below.

Input format:

```
merges
mired
mmiergreeds
```

Output format:

`

VALID

S1**m** S2**mi** S1**erg** S2**re** S1**e** S2**d** S1**s**

Note: More than one valid interleavings are possible.

Q5. **[15 marks]**

An XYZ company is drilling holes in a field to look for minerals. If it finds valuable minerals at a location then it makes a profit, however, if it doesn't find any minerals then it loses money. Assume the field is divided into `nxn` square cells and there is an integer value associated with each cell. A positive integer value indicates potential for profit, while a negative value indicates loss. See illustration below. The company wants to find a square region in which the sum of values of all the enclosed cells is maximized.

1	-2	9	3	4	-8	-3	7
6	7	-8	2	3	1	-4	2
4	-1	4	5	3	-4	5	-1
4	-2	3	-3	1	2	-5	4
6	-1	-2	10	12	4	3	9
4	11	-3	-2	1	1	7	6
9	-5	-1	7	5	3	4	2
-1	1	2	-2	2	3	-1	-3

1. Code an **efficient** algorithm in C/C++ that finds such a square region. Your code will ask the user for positive integer values of 'n'. In your code, provide the following two options: (1) read the values of the matrix from an array (the array will be one-dimensional of size nxn and you will use it to populate your 2-D matrix, (2) randomly fill all the `nxn` cells with positive and negative integer values. The output of your program should be the (1) print the nxn matrix (2) row and column indices of the top left corner and the bottom right corner of the optimal square region, and the width of the square (3) the sum of the cell values enclosed in that square region. [14]
2. Give a clear description of your algorithm and the data structures used to implement it in the comments at the beginning of your code. **What is the running time of your algorithm and how much space does it use?** Your should include clear comments that explain your algorithm's time and space complexity. [1]
3. You algorithm should run for large values of '`n`'. The above example is only for illustration purposes. For testing, remember to change the seed of your random number generator in different executions of the algorithm.

`

Instructions and policies

1. You must submit your ***own*** work. You may discuss the problems with other classmates but must not reveal the solution to others or copy someone's work. Remember to acknowledge other classmates if discussions with them has helped you.
2. You should name your code files using the following convention: Qx-rollno.c
3. Type your answer to the theoretical questions and submit a separate pdf file for each using the naming convention above.
4. Upload all your files in the corresponding assignment folder on LMS. **There will be a 20% deduction for assignments submitted up to one day late (the late deduction is only applicable to the questions submitted late, not on the whole assignment). Assignments submitted 24 hours after the deadline will not be marked.**
5. There will be vivas during grading of the assignment. The TAs will announce a schedule and ask you to sign up for viva time slots. Failure to show up for vivas will result in a **70% marks reduction** in the assignment.
6. In the questions where you are asked to create test cases. Think carefully about good test cases that check different conditions and corner cases. The examples given in the assignment are for clarity and illustration purposes. You should not assume that those are the only test cases your code should work for. Your code should be able to scale up to larger input sizes and more complex scenarios.
7. Do not make arbitrary assumptions about the input or the structure of the problem without clarifying them first with the Instructor or the TAs.
8. Make sure that your code compiles and runs on Ubuntu. You may choose to develop your code in your favourite OS environment, however, the code must be properly tested on Linux before submission. During vivas, your code should not have any compatibility issues. It's a good idea to use gcc -Wall option flag to generate the compiler's warnings. Pay attention to those warnings as fixing them will lead to a much better and robust code.
9. For full credit, comment your code clearly and state any assumptions that you have made. As mentioned in the beginning of the assignment, there are marks for writing well-structured and efficient code.
10. Familiarize yourself with LUMS policy on plagiarism and the penalties associated with it. We will use a tool to check for plagiarism in the submissions.