CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

NON-LINEAR DATA STRUCTURES AND ALGORITHMS.

ASSIGNMENT 2: Dynamic Programming & Backtracking

Part 1 & Part 2: Dynamic Programming (Week 8 – Week 9)

Instructions.

Remember to check the slides of the lectures (the pseudo-code algorithm is in the slides) and to check the code example of the top-down rod cutting. Also, it is advisable to write in paper the algorithm and the draws of the pieces with a small example before starting the coding.

IMPORTANT: If you submit the top-down (recursive) approach you will get 0 marks (only for exercise 1). If you solve the problem using an algorithm that is not Dynamic Programming (such as using only recursion), you will get 0 marks. If you solve the problem with Dynamic Programming for an specific number of *n* but you do not generalize it to any *n* number, you will get only 1 mark.

Part 1: Paper roll cutting (Week 8)

1. Background.

Consider that you are working for a paper rolls company. You have to implement a program that given a paper roll of length *n* computes the best revenue of cutting the roll on several roll-pieces.

There are **ONLY** 4 possible lengths of roll-pieces:

- Roll-piece of length 1 with price 1.2€
- Roll-piece of length 2 with price 3€
- Roll-piece of length 3 with price 5.8€
- Roll-piece of length 5 with price 10.1€

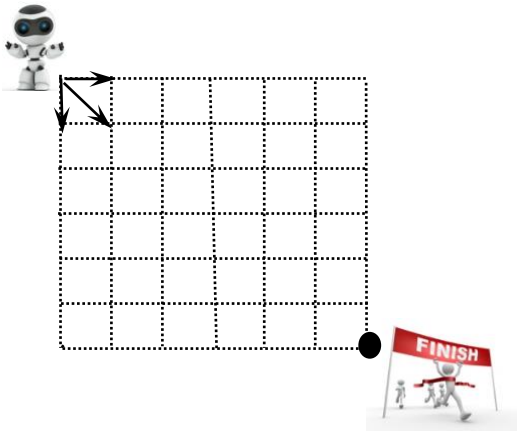
2. Goal of the Assignment.

Implement the dynamic programming **bottom-up (iterative)** approach in Java of the rod cutting problem that we studied in the lectures. The name of your Java file must be: **PaperRollCuttingBottomUp.java**. For any length *n* of a paper roll, the Java file has to print on the screen the best revenue in € and how many roll-pieces of each type were cut for obtaining such revenue.

Part 2 - EXTRA: Robot moving until finish point (Week 9)

1. Background.

You are working for a company that creates robots and you have to code the software for solving the problem of moving the robot until the finish point by consuming the minimum cost of energy for the robot. Please, look the picture below. The robot can ONLY move to the right, down and diagonal-down. As input there is a parameter *n*, which indicates the matrix of *n*x*n* that represents the space. Note that the robot is originally placed in the left-up part of the matrix and the finish point is in the right-down part of the matrix.



The energy costs of the robot moving are: (according to the first letter of your surname):

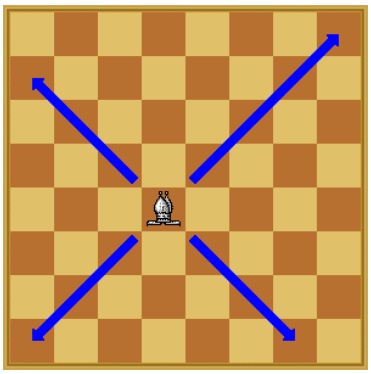
Movement	[A-M]	[N-Z]
Right	1.1	1.5
Down	1.3	1.2
Right-Down	2.5	2.3

2. Goal of the Assignment.

Implement the dynamic programming bottom-up or top-down approach (as you prefer) in Java of the robot moving until reaching the finish point. The name of your Java file must be: **RobotMoving.java**. For any size of the matrix *n*, the Java file has to print on the screen the minimum cost of reaching the finish point and the movements done by the robot until reaching such point (e.g right, down, right-down, right, down, down, down, etc.)

Part 3: Backtracking. (Week 10)

Implement the backtracking algorithm of the bishops chess problem for any number *n* of bishops. In this problem you have to assign *n* bishops to a chess board *(n x n)* so that they are not killing each other. Remember that the bishops only kill in diagonal movements. Print **ALL** the solutions on the screen as chess boards *(n x n)* in which in every cell there is the symbol ***** where there is no bishop and the symbol *B* where there is bishop.



The name of your Java file must be: **Bishops.java**

IMPORTANT: If you solve the problem using an algorithm that is not Backtracking (such as Generate & Test, which generates all the possible combinations), you will get 0 marks. If you solve the problem with Backtracking for an specific number of *n* but you do not generalize it to any *n* number, you will get only 1 mark.

MARK BREAKDOWN.

Assignment 2: 25 marks.

Part 1: 7 marks
Part 2-EXTRA: 8 marks
Part 3: 10 marks

To evaluate the assignment I will run it over some tests on different *n* (run your software and check that the outputs are correct!). Also, as always, remember to print the error messages. The results are based on the performance of your code over these tests.

For each exercise, there are 4 possible scenarios:

A. The function passes all the test, it is efficient and prints error messages → 100% of marks.
B. The function does not pass all tests by small/medium mistakes→ 90% of marks – 10% marks (depending how small is the mistake).
C. The function does not pass all tests by big mistakes, it does not compile or it generates and exception (makes the program crash) or the function was not attempted → 0% of marks.
D. The function works well but you do not pass the demo → 5% of marks.

IMPORTANT: I will use a source code plagiarism detection tool. In case of detecting a copy (for at least one method) between some students, all these students get 0% marks for the whole assignment. Including the student that originally coded it.

SUBMISSION DETAILS.

Deadline: Sunday 11:30.pm of week 11.

Submission Details: Upload to Blackboard **ONLY** the following files:

PaperRollCuttingBottomUp.java.java
RobotMoving.java
Bishops.java

IMPORTANT: Do not ZIP the files.

Lab Demo:

A brief individual interview about the assignment will take place on our lab session on the lab of week 12. **The demo is mandatory for the assignment to be evaluated.**
Please, let me know in the lab if you are willing to do the demo earlier. (With the corresponding uploading of the full assignment to the blackboard). Once the demo is done, there will be no possibility of re-evaluating later. The evaluation will be done with the exact code presented at this moment. Therefore, my advice is to take this option only if you are 100% sure that your assignment is already completed.

If you cannot attend (for important reasons) to the demo, you must talk with me and we will do another appointment to the demo no later than scheduled demo.