CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

LINEAR DATA STRUCTURES AND ALGORITHMS.
ASSIGNMENT 2: ALGORITHMS

BACKGROUND.
In this assignment we are going to implement **divide&conquer** and **greedy**-based algorithms for solving different problems.

Note: The exercises proposed in this assignment are related to the exercises seen in the lectures. Thus, we strongly recommend to download, get to understand, run and debug the code examples of the lectures before start attempting the exercises of the assignment.

ASSIGNMENT 2 – HINT 1
Divide and Conquer: First set of exercises.

BACKGROUND.

The folder **/src** contains the following files:

- **(MyList.java, MyStaticList.java, MyNode.java, MyDynamicList.java):** These classes stand for the package MyList<T> we have seen previously in the lectures of the Block II: Data Structures.
- **DivideAndConquerAlgorithms.java:** This class contains the proposed divide&Conquer functions you have to implement.
- **MyMain.java:** This class tests the functionality of the divide&Conquer functions.

The folder **/doc** contains the documentation of the project. In particular:

- **(MyList.html, MyStaticList.html, MyNode.html, MyDynamicList.html):** Contains the description of the package MyList<T> classes.
- **DivideAndConquerAlgorithms.html:** Contains the description of the class DivideAndConquerAlgorithms.java.
- **MyMain.html:** Contains the description of the class MyMain.java.

EXERCISE.
Implement the following functions of the class DivideAndConquerAlgorithms.java.

1. `public int maxInt(MyList<Integer> m);`
   The function returns the maximum item of m (-1 if m is empty).

2. `public boolean isReverse(MyList<Integer> m);`
   The function returns whether *m* is sorted in decreasing order or not.

3. `public int getNumAppearances(MyList<Integer> m, int n);`
   The function returns the amount of times that the integer *n* appears in *m*.

4. `public int power(int n, int m);`
   The function returns n^m.

5. `public int lucas(int n);`

   Mathematically, the Lucas series is defined as:

$$L_n := \begin{cases} 2 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ L_{n-1} + L_{n-2} & \text{if } n > 1. \end{cases}$$

   Thus, the Lucas series is as follows:

   2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123 ……..

The function returns the n-est item of the lucas series.
Examples: *lucas(0)* → *2, lucas(4)* → *7*

6. `public void drawImage(int n);`

The function prints prints a pattern of a given length.

```
*
**
***
…
```

ASSIGNMENT 2 – HINT 2
Divide and Conquer: Second set of exercises.

BACKGROUND.

The folder **/src** contains the following files:

- **(MyList.java, MyStaticList.java, MyNode.java, MyDynamicList.java):** These classes stand for the package MyList<T> we have seen previously in the lectures of the Block II: Data Structures.
- **DivideAndConquerAlgorithms.java:** This class contains the proposed divide&Conquer functions you have to implement.
- **MyMain.java:** This class tests the functionality of the divide&Conquer functions.

The folder **/doc** contains the documentation of the project. In particular:

- **(MyList.html, MyStaticList.html, MyNode.html, MyDynamicList.html):** Contains the description of the package MyList<T> classes.
- **DivideAndConquerAlgorithms.html:** Contains the description of the class DivideAndConquerAlgorithms.java.
- **MyMain.html:** Contains the description of the class MyMain.java.

EXERCISE.
Implement the following functions of the class DivideAndConquerAlgorithms.java.

7. `public void recursiveDisplayElements(MyList<Integer> m);`
   Given a MyList, this recursive algorithm displays its elements by screen (if any).

8. `public MyList<Integer> smallerMyList(MyList<Integer> m, int e);`
   The function filters all elements of MyList being smaller than 'e'.

9. `public MyList<Integer> biggerEqualMyList(MyList<Integer> m, int e);`
   The function filters all elements of MyList being bigger or equal than 'e'.

10. `public MyList<Integer> concatenate(MyList<Integer> m1, MyList<Integer> m2);`
    The function computes a new lists whose content is the concatenation of m1 and m2.

ASSIGNMENT 3 – HINT 3
Greedy Algorithms

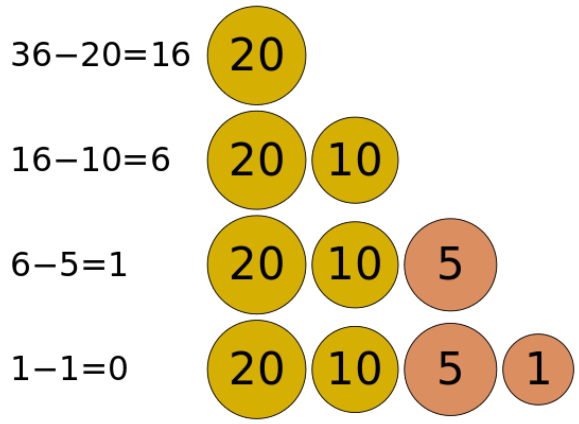
BACKGROUND.

The folder **/src** contains the following files:

- **ChangeMaking.java:** This class contains the proposed Greedy algorithm that you have to implement. The algorithm solves the problem of change making described below.

The **change-making problem** addresses the question of finding the minimum number of coins (of certain denominations) that add up to a given amount of money. It is a knapsack type problem, and has applications wider than just currency.

Greedy algorithms determine minimum number of coins to give while making change. Note that there is no limit on how many coins can be returned from each type of coin. These are the steps a human would take to emulate a greedy algorithm to represent 36 cents using only coins with values {1, 5, 10, 20}:



EXERCISE.
Implement the empty functions of the class ChangeMakingJava.java.

Important: the coins used for the greedy solution must be displayed in the screen in your implementation.

MARK BREAKDOWN.

Assignment 1: 25 marks.

- Hint 1 & Hint 2 → 15 marks
- Hint 3 → 10 marks

To evaluate each function, I will run it over some tests. The results are based on the performance of your code over these tests.

For each function, there are 3 possible scenarios:

A. The function passes all the test → 100% of marks.
B. The function does not pass all tests by small details → 100% of marks – 0.25/0.5 marks (depending how small is the mistake).
C. The function does not pass all tests by big details, it does not compile or it generates and exception (makes the program crash) or the function was not attempted → 0% of marks.

IMPORTANT: I will use a source code plagiarism detection tool. I will also make detailed questions of your code. In case of detecting a copy (for at least one function) between some students or a copy from internet, ALL these students get 0% marks for the WHOLE assignment. INCLUDING the student that originally coded it.

SUBMISSION DETAILS.

Deadline.
Monday 4th of December, 11:00 am.
Submission Details.
Please submit to Blackboard (folder Submissions, A01) **ONLY** the following files: (NO .zip/.rar)

- Hint 1 → File "DivideAndConquerAlgorithms.java".
- Hint 2 → File "DivideAndConquerAlgorithms.java".
- Hint 3 → File "ChangeMaking.java".

Lab Demo.
A brief individual interview about the assignment will take place on our lab session on week 13.
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** than week 13. (With the corresponding uploading of the full assignment to the blackboard in the right deadline). The evaluation will be done with the exact code presented at this moment.
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 week 13.