



LINEAR DATA STRUCTURES AND ALGORITHMS.

ASSIGNMENT 2: ALGORITHMS

IMPORTANT NOTE: The exercises proposed in this assignment are related to the exercises studied in the lectures. I strongly recommended you download, get to understand, run and debug the code examples of the lectures before starting to attempt the exercises in the assignment.

BACKGROUND.

In this assignment we are going to implement divide&conquer (**mandatory to use recursion**) and greedy-based algorithms for solving different problems.

ASSIGNMENT 2 – PART 1

(Week 8)

Divide and Conquer: First set of exercises

BACKGROUND.

The unit on Canvas contains the following files:

- **MyMain.java**: This class tests the functionality of the divide&conquer implementation.
- **MyList.java, MyStaticList.java, MyNode.java, MyDynamicList.java**: These classes define for the package `MyList<T>` which 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.

EXERCISE.

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

1. `public int maxInt(MyList<Integer> m);`
This function returns the maximum item of `m` (-1 if `m` is empty).
2. `public boolean isReverse(MyList<Integer> m);`
This function returns whether `m` is sorted in decreasing order or not.
3. `public int getNumAppearances(MyList<Integer> m, int n);`
This 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);`
This function prints a pattern of `'*'` for a given length.

```
*
**
***
...
```

ASSIGNMENT 2 – PART 2

(Week 9)

Divide and Conquer: Second set of exercises

BACKGROUND.

The unit on Canvas contains the following files:

- **MyMain.java**: This class tests the functionality of the divide&conquer implementation.
- **MyList.java, MyStaticList.java, MyNode.java, MyDynamicList.java**: These classes define for the package MyList<T> which 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.

EXERCISE.

Implement the following functions of the class DivideAndConquerAlgorithms.java.

1. `public void recursiveDisplayElements(MyList<Integer> m);`
Given a MyList, this recursive algorithm displays its elements by screen (if any).
2. `public MyList<Integer> smallerMyList(MyList<Integer> m, int e);`
The function filters all elements of MyList being smaller than 'e'.
3. `public MyList<Integer> biggerEqualMyList(MyList<Integer> m, int e);`
The function filters all elements of MyList being bigger or equal than 'e'.
4. `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 2 – PART 3





(Week 10)

Greedy algorithms.

BACKGROUND.

The **change-making problem** addresses the question of finding the minimum number of coins 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}:

Accuracy:	Coins:	Number Coins:
$36 - 20 = 16$		1
$16 - 10 = 6$		2
$6 - 5 = 1$		3
$1 - 1 = 0$		4

The unit on Canvas contains the following files:

- **MyMain.java**: This class tests the functionality of the greedy changemaking implementation.
- **MyList.java, MyStaticList.java, MyNode.java, MyDynamicList.java**: These classes define for the package `MyList<T>` which we have seen previously in the lectures of the Block II: Data Structures.
- **ChangeMaking.java**: This class contains the proposed Greedy algorithm that you have to implement. The algorithm solves the problem of change making described above.

EXERCISE.

Implement the functions of the class `ChangeMakingJava.java` according to the schema of greedy algorithms studied in the lectures.

1. `selectionFunctionFirstCandidate`:
Basic policy: Just pick the first non-discarded type of coin.
2. `selectionFunctionBestCandidate`:
More elaborated policy: Pick the biggest non-discarded type of coin.

3. feasibilityTest

Given a current solution and a selected candidate, this function states whether the candidate must be added to the solution or discarded

4. solutionTest

Given a current solution, this function states whether it is a final solution or it can still be improved

5. objectiveFunction

This function computes how many coins are used in the solution

6. solve: This function calls to the above functions in order to solve the problem. The selection function executed is `selectionFunctionFirstCandidate` if `typeSelectFunc=1`. Instead, if `typeSelectFunc=2`, `selectionFunctionBestCandidate` is the selected function .

This function is responsible for printing on the screen the output of the problem (see Figure above as example of output) and must include the:

- Accuracy, which is the difference of amount of change provided minus the target amount of money (input of the function).
- Coins that are provided as change.
- Number of coins provided as change.

MARK BREAKDOWN.

Assignment 2: 25 marks.

- Part 1: 9 marks (1.5 marks each function)
- Part 2: 6 marks (1.5 marks each function)
- Part 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. Remember to print the error messages and comment your code.

For each function, there are 4 possible scenarios:

- A. The function passes all the test -> 100% of marks.
- B. The function does not pass all tests by small/medium mistakes -> 90% of marks – 10% marks (depending how small the mistake).
- C. The function does not pass all tests due to big mistakes, it does not compile or it generates an 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 because you do not answer my questions about your code or your answers are wrong -> 5% of marks.

IMPORTANT: I will be using a source code plagiarism detection tool. In the 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 6th of December, 11:59pm.

Submission Details.

Please submit the following on Canvas:

- Part 1 File “DivideAndConquerAlgorithms.java”.
- Part 2 File “DivideAndConquerAlgorithms.java”.
- Part 3 File “ChangeMaking.java”.

Lab Demo.

A brief individual interview about the assignment will take place in your lab time on week 11 (Monday 7th - Friday 11th of December).

This demo is mandatory for the assignment to be evaluated.