# Mandatory Implementation Exercises

The 02105+02326 DTU Algorithms Team

## Exercises

You must pass at least 2 out of 4 of these implementation exercises. To pass an exercise you must upload your solution to CodeJudge and get a green smiley. You may program in any of the languages Java/C/C++/C#/Python/Rust/Pascal.

**M Mandatory Exercise: Implement Pseudocode** Translate the following pseudocode into a program.

```
IntegerAnalyzer()
A = IntArray(ReadInt())
for $i = 0$ to $n-1$ do
  A[i] = ReadInt()
end for
Sort(A)
for $i = 0$ to $\lfloor (n-1)/2 \rfloor$ do
  $s = 0$
  for $j = 0$ to $i$ do
    $s = s + A[j] + A[n-j-1]$
  end for
  PrintInt(s)
end for
```

*IntArray(n) creates an integer array of size n. ReadInt() reads an integer from standard input. PrintInt(s) prints s to standard output. Sort sorts an array in ascending order (use the built-in function in your chosen programming language).*

| Sample Input | Sample Output |
|---|---|
| `5`<br>`3 9 42 -43 32` | `-1 34 52` |

**M Mandatory Exercise: Recursion** Implement a program that reads an integer $n$ from standard input and then prints $f(n)$. Your program should be recursive. $f(n)$ is given by:

$$f(x) = \begin{cases} i & \text{if } i \leq 2 \\ 2f(i-1) + f(i-2) - f(i-3) & \text{otherwise} \end{cases}$$

| Sample Input | Sample Output |
|---|---|
| `5` | `25` |

**M Mandatory Exercise: Alternating Paths** Consider a $n \times n$ grid consisting of 0's and 1's. Create a program that computes the length of a shortest path of alternating 1's and 0's from the upper left corner to the lower right corner (a path can go left/right/up/down). The grid should be read from standard input, the first line is $n$ and the remaining lines are the grid. The program should output the shortest possible length to standard output.

| Sample Input | Sample Output |
|---|---|
| 5<br>00010<br>11111<br>01000<br>01111<br>00000 | 13 |

**M Mandatory Exercise: Binary Trees** The following pseudocode constructs a binary tree from some input:

```
ReadBinaryTree()
$A$ = ReadInt()
if $A = 0$ then
    return NULL
else
    return NewNode($A$, ReadBinaryTree(), ReadBinaryTree())
end if
```

ReadInt() *reads an integer from standard input.* NewNode*(k, l, r) creates a new binary node with key k and left child l and right child r.*

An example of input to the program could be "5 3 0 0 4 2 0 0 1 0 0". Before solving the rest of the exercise, you should try to draw the binary tree resulting from running the program on this input (you don't have to hand-in this drawing).

Implement the above pseudocode. Extend the program to do a pre-order traversal of the binary tree. When visiting a node $v$ in the traversal, you should print the sum of $v$'s, LEFT($v$)'s and RIGHT($v$)'s keys (assume the key of a null node is 0, but don't print anything for NULL nodes).

| Sample Input | Sample Output |
|---|---|
| 5 3 0 0 4 2 0 0 1 0 0 | 12 3 7 2 1 |