Object-Oriented Programming

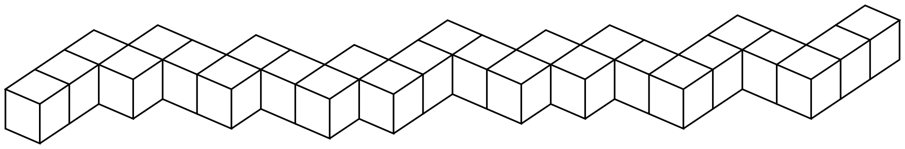
Programming Project

Academic Year 2023-2024

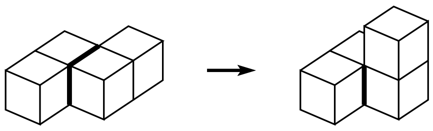
This project can be completed individually or by pairs of students, and must be submitted by email to `bvergain@uliege.be` *for April 14th at the latest. Projects returned after the deadline will not be graded. Except for classes present in the Java Class Library, which you are allowed to use, the entire code of your project must be written by yourselves. All cases of fraud will be prosecuted according to the policy of ULiège (cf.* `https://www.student.uliege.be/cms/c_11187649/`*). This include copying or adapting code obtained from other students, from online sources, or generated by AI tools.*

The snake cube puzzle

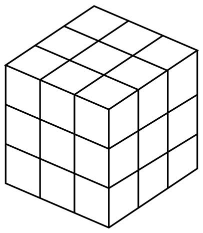
The *snake cube* is a 3D puzzle composed of a chain of n^3 wooden cubes linked together, for some $n \geq 1$, with pairs of consecutive cubes sharing a common face. Initially, all the cubes lay flat on a plane. An example of such an initial configuration is shown below, for $n = 3$:



The shape of the puzzle can be modified by rotating adjacent cubes along their common face, as shown below. (The edges of the rotation face are drawn in bold.)



The goal of the puzzle is to find a combination of rotations that packs the n^3 cubes into a big $n \times n \times n$ cube.



Subject of the project

The project consists in writing a Java program that is able to solve the snake cube puzzle, by starting from a given initial configuration, and searching for a valid combination of rotations.

A character string encoding the initial configuration will be provided as an argument to the program, in which each successive cube of the chain is described by a letter corresponding to its geometry:

- `E` (for *Endpoint*) if the cube is the first or the last one, i.e., it has only one neighbor (or zero if $n = 1$).
- `S` (for *Straight*) if the cube has two neighbors on opposite sides.
- `A` (for *Angle*) if the cube has two neighbors that share a common edge.

For instance, the puzzle configuration in the first figure is encoded by `"ESAAASAASAAASASAAAASASASASE"`.

If a solution is found, the program must display it by outputting sequentially and on separate lines of text the position of all the n^3 cubes in the chain. The position of a cube must be given as a triple (x, y, z) in a 3D coordinate system, such that $x, y, z \in \{1, 2, \ldots, n\}$. For instance, the $2 \times 2 \times 2$ snake cube with the initial configuration `"EAAAAAAE"` admits (in particular) the solution

```
(1, 1, 1)
(2, 1, 1)
(2, 2, 1)
(1, 2, 1)
(1, 2, 2)
(2, 2, 2)
(2, 1, 2)
(1, 1, 2)
```

Important guidelines

- Your program must expect to be invoked with a single argument containing the encoding of an initial configuration of the puzzle. It must immediately start searching for a solution and then, if one is found, display it in the format described in the previous section. If the puzzle admits several solutions, only one (selected arbitrarily) must be displayed. If no solution can be found, the program must display `"No solution"`. If the initial configuration is missing or is not correctly encoded, the program must display `"Invalid input data"`.
- It must be possible to start your program by the command `"java SnakeCube"` followed by the initial configuration, for instance

  ```
  java SnakeCube ESAAASAASAAASASAAAASASASASE
  ```

  You are strongly encouraged to check that your program works as expected on this particular example.
- Your submission must take the form of a `.zip` or `.tar.gz` archive containing the Java source code of your project. The name of this archive must be formed by the student IDs of the members of your team separated by dash symbols, for instance `"s2312345-s2054321.zip"`. This file must be attached to an email bearing the subject `"OOP project submission"`.
- After extracting the contents of your submitted archive in the current directory, it should be possible to compile your project with the command

  ```
  javac SnakeCube.java
  ```
- Your project will be evaluated with Java 8. It is thus important to make sure that your submission does not rely on features introduced in later versions of the Java language.
- The evaluation will take into account not only the correct implementation of the requirements of this problem statement, but also the compliance of your source code to the principes of object-oriented programming, as well as its modularity, efficiency, readability, simplicity, and portability.