Team Ids	Grade	Feedback
s099338	6	Your program does not work as expected. It outputs invalid solutions to puzzle that have no solution, and returns « no solution » when there exists some. From what I have seen in your code, the method sort() is incorrect as it only shifts some cubes coordinates, and not every cube (as it should). The method coordinateIsUnvalid(…) is also incorrect: for n=3, you check whether -3 < {x, y, z} <3 when you should test -1< {x, y, z} <3, because you start in (1,1,1), and not (0,0,0).
s110719	14	Your program fails to solve the 1x1x1 snake, but otherwise works pretty well, although it fails to solve one 4x4x4 instance. Try to only submit what works and clean your code a bit (avoid leaving comments like « //XXX Why did I use this again ?? », or classes that are not actually used in your project. Overall, your code is a bit complicated, and it would have helped to use more abstractions (like a class Coordinates in place of int[][] for instance) to ease the reading. There is no reason for some methods (in Chain and Puzzle) to be package private.
s183601	8	You do not detect invalid Snakes with an encoding containing incorrectly placed 'E'. Because of how you compute the size of the cube, your program also fails to solve 4x4x4 cubes. Be careful of the approximations when working with doubles or floats. Furthermore, your display never prints the coordinates of the first cube, and some cube coordinates are repeated in the (incomplete) output solution. The code is however well structured, with some effort on modularity.
s190952-s191962	20	The program works exactly as expected. The code is simple and well commented.
s191876	7	Your project does not work as intended. It provides invalid solutions of the puzzles, and even finds solutions for unsolvable instances. The code is however well structured, with a good effort on controlling the visibility of the methods and variables, and commenting.
s192379	6	As you mention in your code, your program is incomplete and does not work as intended. The code you submitted is, I think, a good first step towards a complete solution.
s192584-s173899	18	On invalid input, your program should actually display "Invalid input data" (as you do when no input snake is given), and not throw an exception containing this text. Your program also fails to solve the 1x1x1 snake, but otherwise works well, except that the positions of the cube are printed in reverse order (you start by the last one). The code is a bit complex, and the commenting is not helping much. Try to submit a clean code with useful comments for an external reader. The method ModifyPosition(…) should not be package private, nor should it start with an upper case.
s193674-s2304031	13	Your program fails to solve 3x3x3 instances, when the solution requires the first cube to be somewhere else than (1,1,1). Otherwise the code is clear and easy to read. You break the encapsulation principle with the public methods allowing to modify the Snake and the Cube freely. In the class Cube, if you want to compare two cubes using the method equals(), make sure you override the method inherited from Object (and also override hashCode()) or try to name your method differently.
s195949-s190410	17	Your program fails to detect when the input snake has an incorrect size, or when no input snake is given. Otherwise, it performs well, except for when the first cube cannot be in position (1,1,1). Otherwise the code is very clear and simple.
S201020	4	Your program does not work at all, it only prints many "No solution" in a row on basically every input. There is some structure in the code that can lead to a working project I believe.
s201369-s202080	11	Your program fails to solve the 1x1x1 snake, as well as some solvable 3x3x3 and 4x4x4 instances. The code is well structured, which should help correct these errors.
s201528-s202524	20	A good project, working as expected. The code is simple and easy to read.
s201585-s2300486	15	You were asked to display "Invalid data input", and not throw an exception containing this text. Otherwise your program works well on positive input (Snakes that admit a solution), but crashes on negative inputs because of a StackOverflow, without detecting the absence of solution. When stuck, your program cannot escape the recursive calls from findSolution() and findOtherSolution(), to each other. The code is well commented, easy to read, and has good structure. Be careful with the visibility of the variables (e.g., in the class Position).
s201766-s202160	14	Your program fails to find a solution for some 3x3x3 solvable snakes, and sometimes give invalid solution (where two consecutive cubes are not adjacent in their final coordinates). The code is however well structure, with good use of inheritance, and good modularity. This should help correct these bugs.
s202250-s200986	13	You were asked to display "Invalid data input", and not throw an exception containing this text. Your program works somewhat well on positive inputs (puzzles that admit a solution) although sometimes it remains stuck. But it systematically crashes on negative input (throwing an IndexOutOfBoundException), without clearly detecting the absence of a solution. The variables in Cube should not be public. The code is otherwise quite clear and easy to read.
s203705-s200502	20	The program works well. The code is clear and simple, I would only argue that setting the visibility of the classes Cube and Coordinates to package private (especially since the attributes of Coordinates are public) is not really justified.

s210477-s221672	19	You were asked to display “Invalid data input”, and not throw an exception containing this text. Your program also fails to detect when the snake encoding contains an ’E’ in the middle. The code is clear and shows some mastery of Object Oriented Programming principles.
s210844	5	Your program does not work as expected, it fails to find solutions for many solvable input snakes. You use very little of what Object Oriented Programming offers, having basically only a single class that does everything (the class Cube is not in a dedicated file, and has no methods). There is no reason for the method solve(…) to be public since you can only call it in the class itself.
s210875-s2307050	12	Your program works seems to work as expected, although being very slow. It struggles on most 3x3x3 inputs. The structure and visibility within your main class Cube is quite odd. You define setters and getters that only this class will use (since its behavior is very specialized for this project) which are not necessary, and set every method as public although being only used in the class itself (apart from solve(...)).
s210880-s212585	5	Your program returns “No solution” to every input snake I tested it on (except the 1x1x1 snake) even when they admit a solution. Your code could use more commenting, especially describing what you expect (precisely) each method to do. This is crucial in order to debug your project. One issue for instance is that you have a comment saying “If any of the cubes are on the same position, return true” in a loop where you return false if they share the same position.
s210900-s203699	20	The program works exactly as expected. The code is clear and well commented. The only real issue is that you print the coordinates of the cubes in reverse order (you start by the last one).
s211381-s211885	6	As you mention in your code, your program is incomplete and does not work as intended. Your main class Puzzle has a lot of attributes, make sure you actually need all of them, and maybe try to design dedicated structures to group some of them. It will make it easier to understand and debug your code. Also, make sure that the final display actually matches what is asked in the statement, otherwise it makes testing your code even more complicated.
S211704-S210801	13	You were asked to display “Invalid data input”, and not throw an exception containing this text. Similarly, on some cases where the input snake admits no solution, an exception is thrown instead of printing the text on the standard output. There are also some solvable snakes for which your program is not able to find a solution.
s211757-s221594	20	The program works well and quite efficiently. The code is clear and very well documented.
s211883	18	Your program works well except when the solution requires the first cube to be in another position than (1,1,1). You actually mention that in your code, but it is also contradicted a few sentences before, when you say: “So when it happens, we moove the first cube to the next non equivalent position.” which you do not. Such a change would not really complexify your code, which is very clear, well structured, and well documented.
s212097-s216862	8	Make sure that your class names start with an upper case letter, and only give .java files in your submission. If you override equals(), you must also override hashCode(). Your program breaks on many instances. The method solveRecursive(…) seems to be at fault, through its recursive calls, the value of cubeRoot somehow increases which forbids your program from cleanly exiting the loops.
s213330-s211764	20	Try to follow the statement guidelines for the user interactions, otherwise it makes it harder to correct your project. Otherwise your program works as expected. The code is clear, and well documented, with a good use of inheritance.
s213455	9	Your program fails to find solutions for many solvable 3x3x3 instances, apart from the one from the statement. Also, why return (0,0,0) instead of (1,1,1) for the unit snake ‘E’? There is a significant effort to make the code modular, but the class GeometryUtils seems overly complex. It is usually not recommended to work with doubles when you could just work with integers the whole time.
s214109	20	The program works exactly as expected, and is very efficient. The code is clear and well commented.
s215116	20	The program works well and is quite efficient. The code is very simple and well documented.
s215439	17	Your code does not compile under Java 8. You use the operator List.of(…) which was only introduced in Java 9. Otherwise, (with this issue corrected) your program works well and quite efficiently. The code is clear and modular. If you define a method equals() in your class, it must take an Object to override the equals() inherited from the class Object, and you must then also override the method hashCode().
s217043	3	Your code simply does not solve the puzzles. It only gives a way to fill the NxNxN cube without looking at the configuration of the input Snake. The code does not respect the principles of Object Oriented Programming as it contains a single class where everything is static.
s218152-s225849	20	The program works as expected. The code is well structured and is well commented.
s220252-s224642	8	Your program gets stuck on some clearly unsolvable inputs, and give invalid solutions to some solvable and unsolvable ones. Furthermore, you print the cube positions in reverse order (you start by the last one). Although it does not correctly solves the problem, your code has a clear structure, and there is a real effort commenting it. This is a good thing to help you debug your project.

s220622	12	You do not consider snakes with invalid length as Invalid Input Data, but you fail to solve the 1x1x1 snake. Otherwise your program works well. Your code does not really comply with Object Oriented programming principles. You do have multiple classes, but you do not instantiate any of them, you simply use them as collection of separated static methods. For this reason you end up with ArrayList of ArrayLists of ArrayLists of Integers in your code which is not so convenient to manipulate. It works for this project, but would be unsustainable at a larger scale.
s220801-s212115	17	Your program works quite well, except when the solution requires the first cube to start somewhere else than in (1,1,1). It is otherwise quite efficient. The code is clear and well structured.
s220817-s221606	16	Your program works quite well, except for the 1x1x1 snake, or when the solution requires the first cube to start somewhere else than in (1,1,1). It is otherwise quite efficient. The code has good modularity, is well commented and easy to read.
s220820-s221453	6	Your program does not work as expected. It should not throw an Exception when the input is invalid, but only display “Invalid input data”. Furthermore, you sometimes find solution to unsolvable instances, and most of the solution you give are clearly invalid, as consecutive cubes do not have adjacent coordinates. Your code has however a coherent structure, and is rather well documented.
s220971-s225857	14	Your program does not solve the 1x1x1 snake, it also does not seem to consider other starting position than (1,1,1) for the first cube. The fact that your recursive calls generate such Stack Overflow is problematic. Your program should perform a depth-first search of a solution instead of a breadth-first search to avoid this issue. The code is quite clear and well structured.
s220988	20	Your program works as expected and quite efficiently. The code is simple, well structured and well documented.
s221009	8	The program does not finds solutions for many solvable input snakes, and in particular the one given in the statement. Although you do acknowledge the fact that your program does not work for 3x3x3 puzzles and further, which is a good thing, it could have been stated more clearly. The code is however rather well structured and there is a real effort on commenting, which should help you debug it. Your definition of the InvalidInput exception class is not good. You should not store another string for the error message, an existing method inherited from Exception does that already.
s221278-s224697	19	Your program does not behave well on some invalid input, either displaying “No solution” or even getting stuck when symbols other than ‘A’, ‘E’, or ‘S’ are involved. Otherwise, it works well and quite efficiently. The code is well structured, with good modularity and well commented.
s221281	15	Your program behaves well, except that it seems to give wrong solutions to puzzles that require the first cube to start in another position than (1,1,1): it places the first two cubes at the same coordinates. The code is easy to read, but could be more modular and delegate more work outside of the main class SnakeCube.
s221348	16	Your program behaves very well and quite efficiently. Your code is well structured, but you need to be careful with the visibility of your classes and their attributes. The attributes should basically always be private, and it makes little sense to have the enum Direction package private, but its methods public.
s221501	19	Your program should not throw an exception on invalid input, but only print “Invalid input data”. Otherwise your program works well and is efficient. The code is well structured and commented.
s221723-s193784	5	Your project is not organized as required by the statement, the main method should be in the class SnakeCube, not in another Main class. Furthermore, the main() function simply does not compile. This is easy to correct, but you should never submit a project which does not compile or that cannot be executed as described in the statement. Your program does not detect invalid input snakes, or solve the 1x1x1 snake. It behaves correctly on the 2x2x2 instances, but seems to get stuck on the 3x3x3 instances. It also looks like you mixed two different projects into a single one, but without using both parts (the classes Cube2 and Snake are actually never used apart from the first part of the main() method where it does something that is not asked, and are somewhat redundant with Cube and SnakeCube).
s221838-s225222	20	Your program works as expected and quite efficiently. The code is not commented but clear enough to be easy to read. If you implement a class inheriting from Exception, try to give it a name with “Exception” and not “Error”.
s221843-s225150	19	Your program works as expected, although being a bit slow. The code is very modular and well documented.
s221965-s225089	4	Your program should only print the solution or the error messages, not a description of the run. Otherwise, it does not really work. The 2x2x2 instances are not dealt with correctly, as you return an invalid solution to solvable instances, and find solutions to unsolvable ones. The 3x3x3 instances are not solved as they generated Stack Overflows. There should be a dedicated file for each of your classes.
s221988-s221148	16	Your program does not consider solutions where the first cube is not placed in (1,1,1). Otherwise, it works fine. The code is clear and well structured, although the class Tools is a bit superfluous, and its only method isPerfectCube() should be static: there is no reason to instantiate the class Tools.

s222315-s221782	20	Your program works as expected and quite efficiently. The code is clear, well structured and well documented.
s222337	10	Your program is (as you mention in the ReadMe file) very slow. It seems to work on 2x2x2 instances although the final coordinates are not contained within 1 and N as requested. Your program also crashes instantly on 4x4x4 instances, because you increase the variable nb_face with the size of the snake, which does not make sense since you want to obtain a cube in the end, the number of possible rotations is constant regardless of the size of the input. Your code is however rather well structured and commented. The methods in the class Matrix should all be static, there is no reason to instantiate this class.
s222343-s214756	8	Your program should not throw an exception on invalid input, but only print “Invalid input data”. Furthermore, it finds solution to unsolvable instances, and return invalid solutions to solvable ones (the final coordinates do not match the structure of the input snake). The code is however well organized and commented, which should help you debug it.
s222344-s213526	18	The program works as expected, and is quite efficient. Globally, the code is rather well structured and well commented. The Coordinate variable in Cube should be private and not package private. The classes DoList and Position are a bit superficial, the two static methods they contain could have been moved to the class Solve for instance.
s222350	20	The program works as expected and quite efficiently. The code is clear, well structured and well documented.
s222643-s211562	18	Your program works well, except that it get stuck when the input snake has an invalid size. The code is clear and well structured.
s222829-S223469	16	Your program works well, but the format of the solution is very different from the one described in the statement. It makes especially little sense to display a specific configuration when your program finds no solution. The code structure globally satisfies the Object Oriented Programming principles.
s223751-s216284	7	The program’s output is very different from the one described in the statement. Furthermore, most solutions displayed are clearly invalid, one cube is placed at (13,14,2) for a 3x3x3 puzzle. The code is not well structured. For instance, you could have separated the part that deals with the encoding, from the part actually solving the puzzle. Why is the instance variable snake not private in SnakeCube when all others are?
s224003-s225236	19	The program behaves as expected and quite efficiently. The code is very short and clear, although it is always better to have some comments.
s224318	18	Your program works as expected and quite efficiently. The code is clear and simple. It could nevertheless contain some comments. And it should not consider 5*5*5 as the maximum size for an input snake.
s224331-s221065	18	Your program works well and is quite efficient. The code is modular, but be careful with the visibility of your variables. The class variable n in Cube being static does not protect it from being altered by any class within the package.
s224385-s142720	3	Your program should only display what was described in the statement. Your submission was very annoying to test with all the printing mixed with the resolution. The resolution should be separated from the way you return the solution in the end. Furthermore, the coordinates of the solutions are not contained within 1 and N as required. A lot of valid input snakes are considered invalid (the 1x1x1 snake, the 4x4x4 snakes), and for the ones considered valid, it either finds solution to unsolvable puzzles, or gives invalid solutions to the solvable ones. The code has however some structure, and you put some efforts into commenting.
s224395	13	Your program works as expected, except that it crashes on the 1x1x1 snake. The method isCube() should be static, there is no reason to actually instantiate the class IsCube. The class Block could have inherited from Coordonnee, instead of basically duplicating the setters and getters methods. The class SnakeCube could have delegated a more work to other classes.
s224689	15	Your program works almost as expected. When the solution requires the first cube to be in another position than (1,1,1), you try to solve the puzzle again from another initial position, but the Puzzle you work with has already been altered in the previous search (it is most likely the variable positionsVisitees that is at fault). This prevent your program to actually find a valid solution. The code is clear and well structured. There is however no reason for the variables isValidConfig and isValidSize to be package private. They should be either private or final.
S224691-s222349	18	Your program behaves as expected, except for the 1x1x1 snake, and when no input is given to the program. The code is well structured and easy to read.
s224910-s222347	20	Your program works as expected, and is quite efficient. The code is clear and very well documented.
s224919	17	Your program works as expected, except when no input snake is given. The code is clear and work is delegated among the classes. Be careful however when you override the method equals(), it must take an Object as its argument, and you must also override the method hashCode().

s225055-s224596	11	Your program sometimes fail to find a solution to solvable instances, or give invalid solutions (for one of them, the first cube was placed at (0,1,2) for instance). Furthermore, there was too many nested recursive calls to solve most 3x3x3 puzzles without increasing the Stack memory. Regarding the code, it is rather well structured. However, it is a bit odd to have Axis inherit from Vec3 when they are actually quite different. Also, do not forget that when you override the method equals() in a class (as you do in Vec3) you must also override hashCode() accordingly.
S226180	5	You did not follow the statement requirements that required the main method to be in the SnakeCube class, not in another Main class. Your program does work. It does not find any solution for every solvable instances I tested it on (apart from the 1x1x1 snake). It also fails to detect many invalid inputs.
s227629	15	Your program fails to solve the 1x1x1 snake, as well as instances that requires the first cube to be elsewhere than (1,1,1). The code is well structured. It lacks comments, especially in SnakeCubeSolver where the important work is done.
s2306324-s223035	17	Your program fails to solve the 1x1x1 snake and to clearly identify some invalid inputs, but otherwise it works well and solves correctly the puzzles. The code is well structured. The instance variable gameEnvironment in CubeSolver should either be final, or private.
s2306485-s220095	9	You did not follow the statement requirements that required the main method to be in the SnakeCube class. Furthermore, your program fails to find solutions for many solvable instances. The methods are not too big and well commented, which should help you debug your code.
s2309342	1	Your program does not solve the problem at all, it only outputs a sequence of coordinates that fills the NxNxN cube, regardless of the configuration of the input snake. Your code does not respect the principles of Object Oriented programming at all.
s20217327	0	Incomplete project that does not compile.
s2306965	0	Incomplete project that does not compile.