Algorithm and Data Structures Projects

Instructions

The project consists of five algorithms to be developed in Java.

Methods of carrying out the projects

The use of algorithms and data structures defined in the Java standard library is allowed**.**

Programs must be built as command line applications. Each must be implemented in a single source file called AlgoritmN.java (example: Algorithm1.java, Algorithm2.java, etc). The file must have a public class called AlgorithmN, containing the static method main (); other classes, if necessary, can be defined within the same file. Programs **don't have to specify the package** (so **they don't have to contain** the header "package Algorithm1;" or similar).

Programs must start with a comment block containing any information useful for evaluating the program (for example, considerations on the use of particular algorithm or data structures, asymptotic costs, etc.).

Programs will be compiled from the command line using Java 11 (OpenJDK 11) with the command:

```
javac AlgorithmN.java
```

and always run from the command line with

```
java -cp . AlgorithmN any_input_parameters
```

Programs must not require any further user input. The result must be printed on the screen, **scrupulously** respecting the format indicated in this document, because the programs will undergo a first phase of semi-automatic checks. **Programs that produce non-conforming output will not be accepted.**

It can be assumed that the input data is always correct. Some examples of inputs are provided for the various Algorithm description, with the respective expected outputs. **A program that produces the correct result with the supplied input data is not necessarily correct.** The delivered programs must work correctly on *any* input: for this purpose they will also be tested with inputs other than those provided.

Some problems may admit more than one solution; in these cases - unless otherwise indicated in the specification - the program can return any one, even if different from the one shown in the text or supplied with sample input / output data.

In the case of exercises that require the printing of the results of floating point operations, the results obtained may vary slightly according to the order in which the operations are carried out, or based on whether the data type float or double; **such small differences will be ignored**.

The input files assume that real numbers are represented using the period ('.') As a separator between the integer and the decimal part. This setting may not be the default in your Java installation, but just insert the call at the beginning of the main () method :

```
Locale.setDefault (Locale.US);
```

to set the separator correctly (import java.util.Locale to make the method available).

Additional requirements

The correctness of the proposed solutions must be demonstrable.
Proof that the program is correct may be required when discussing projects.
Smoky arguments that merely describe the program line by line and more are not considered proofs.

The code must be legible. Incomprehensible and poorly structured programs (for example, containing methods that are too long, or an excessive level of nesting of loops / conditions - "if" inside "if" inside "while" inside "if" ...) will be heavily penalized or, in cases more serious, refused.

Use appropriate names for variables, classes, and methods. Using inappropriate names makes the code difficult to understand and evaluate. The use of deliberately misleading identifier names may be heavily penalized when evaluating the entries.

***Comment the code appropriately.* Comments should be used to succinctly describe the pain points of the code, not to paraphrase it line by line.**

Example of useless comments	***Example of an appropriate comment***

v = v + 1; // increment v
if (v> 10) {// if v is greater than 10
   v = 0; // set goes zero
}
G. Kruskal (v); // run Kruskal's algorithm
	// Find the position i of the first value
	// negative in the array a []; at the end we have
	// i == a.length if none exist
	// negative value.
	int i = 0;
	while (i <a.length && a [i]> = 0) {
	   i ++;
	}

Each method must be preceded by a comment block that briefly explains the purpose of that method.

Length of lines of code. The source lines must have a limited length (indicatively less than or equal to 80 characters). Too long lines make the source difficult to read and evaluate.

Use adequate data structures. Unless otherwise indicated, the use of data structures and algorithms already implemented in the JDK is allowed. Deciding which data structure or algorithm is most appropriate for a given problem is one of the objectives of the tasks, and therefore will have a significant impact on the evaluation.