

1 Instructions

Due Date: April, 28th, 2022 no later than 11:59pm.

Task: For the second project,

- Read carefully the project below, and download the template.
- Solve the project, starting from the template. Make sure you test your program thoroughly, with relevant test cases, and comment your code.
- Share the project, zipped, with me on D2L. Remember to write your name and the date in a delimited comment at the beginning of the source code. Make sure that your project is properly uploaded before *04/28, 11:59pm*.
- I am happy to provide partial feedback before the due date. Either upload your current code to D2L and send me an email, or bring the code to me during office hours or an appointment. **DO NOT** seek help from your classmates, the UCAs, or Programming Friday Mornings for this project. Also, make sure all of your code is something you wrote yourself, using your understanding of the instructions. **DO NOT** use a solution found online. You are, however, welcome to look at the official C# documentation or other sources in order to better understand methods and features of C# used in the project.

Knowledge: The project may require an understanding of the following topics:

- Classes and objects,
- Loops,
- Decision structures,
- Arrays¹,
- and Random.

Some of these topics have yet to be covered in class, but all will be covered at least one week before the deadline. I highly recommend reading ahead so that you can complete your project well in advance.

2 Rover On The Line Project

In short: In this project, you are asked to design, implement and test a class that represents a rover navigating on a (finite) line, looking for its target, placed at an unknown location on this line. You will then have to design an algorithm that sends instructions to this rover to find the target and stop on it. Optionally, you will design and implement a way of returning the rover to its original location.

In more detail: Design: You need first to design a class for the RoverOnTheLine problem. This class should contain all the attributes and methods required to represent the following information and actions:

- A rover only explores a finite, “1-dimensional”, ground (think of a segment on a line).
- The initial position of the rover and the position of the target is on this finite ground.
- The number of steps made by the rover so far (that is, the number of times it moved) is recorded.
- The class requires two constructors:
 - A constructor that takes multiple arguments, and sets all the attributes to the arguments, provided they satisfy the specification above (you are free to decide what happens if not). The number of steps, in particular, should be 0.
 - A constructor that does not take arguments, and sets all the attributes to random values: the size of the ground to cover should be a random value lesser than 101, and the other attributes should be set randomly, but respecting the specification above.
- A method to move left, that should move the rover left (that is, decrement its position) and return true if the movement was possible (if the rover would not “fall off the ground”), and leave the position unchanged and return false otherwise.
- A similar method to move right.
- A method that indicates if the rover is on the target.

¹Even if they are not required: you can solve this problem without using arrays. It may even be simpler to do without.

- A method that returns a description of the current state of the rover.

Implementing: Once you have all the elements you need to implement this class, start implementing it. “Mock-up” methods are included in the template, but you should edit their signature, change their body, and tweak them any way you see fit. Only the names are fixed. Once you have completed your implementation, the program in Program.cs should behave “as expected” and create a Rover on a ground of size 5, place the rover on its right-left end, and put the target at 2. It will then display information about the rover, try to move it right (but fails to do so), move it left three times, check that it is indeed now at the same location as the target, and then displays the information one more time.

Testing: As a final test, you need, in the Main method, to

- Create a rover with random values, using your no-arg constructor,
- Write a small program that guarantees that your rover will always find its target and stops.
- Displays the total number of steps made by the rover.

Optional: Edit your class and program so that there is also a GoBack method that returns the rover to its original position.

Example: Here is a sample execution (where the optional part was completed: you need not to display the original position unless you are planning on completing this part as well).

```
Ground to cover   : 1..5
Current Position  : 5
Original Position : 5
Target           : 2
Number of steps   : 0
```

You cannot move right!

You found it!

```
Ground to cover   : 1..78
Current Position  : 58
Original Position : 43
Target           : 58
Number of steps   : 435
```

You found it!