NFI coding assignment

This assignment asks you to implement a validation program that checks whether a schedule is valid. A schedule contains the following data:

1. A set of 1 or more robots. Each robot has an origin point, which is an (x, y) coordinate.
2. A set of 1 or more time steps. In each time step, each of the robots will visit a specific (x, y) position.

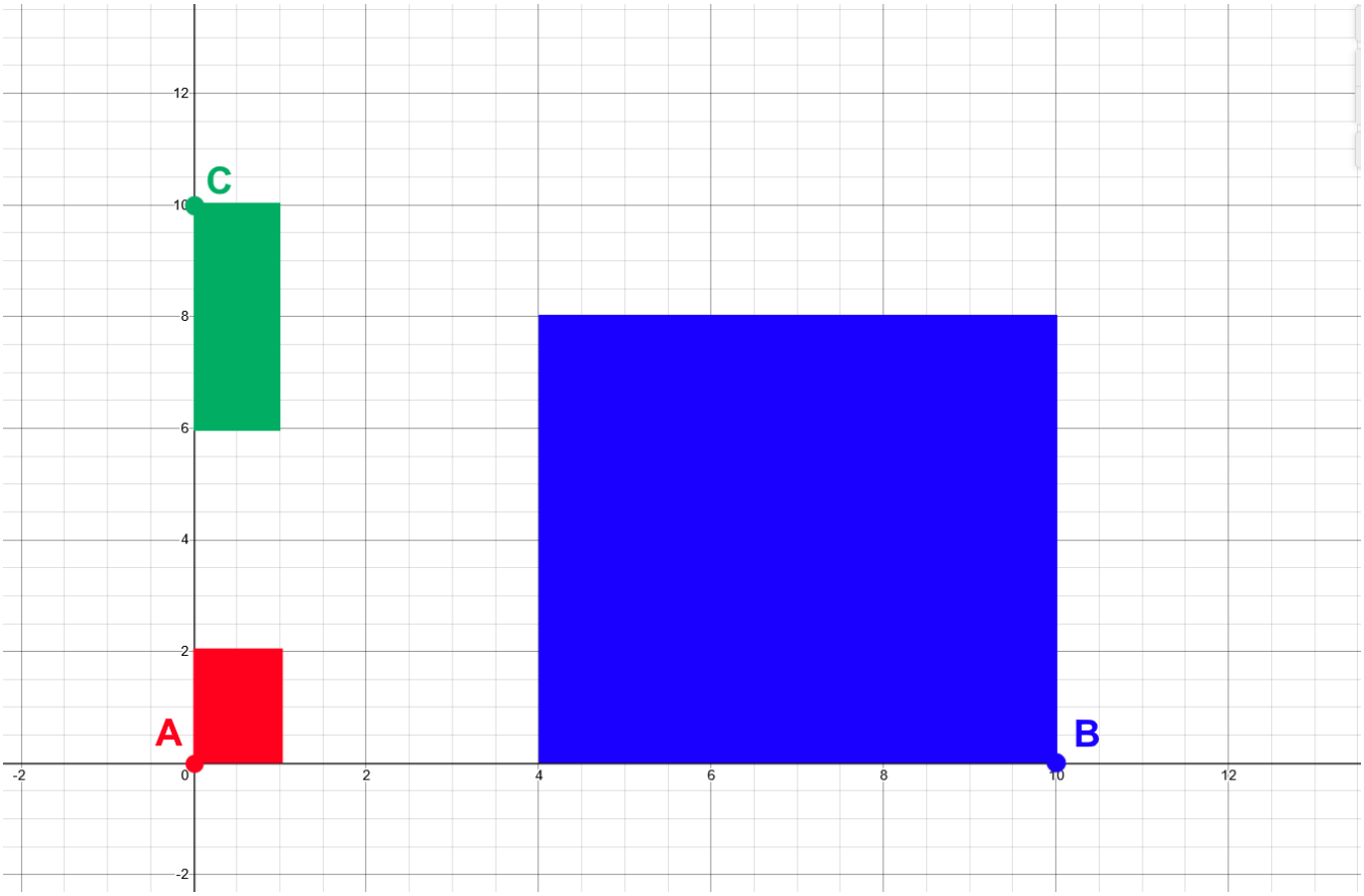
For example, a schedule with 3 robots and 2 time steps could look as follows:

```
# The number of robots
3

# The number of time steps
2

# The robots and their (x, y) origins. Format:
# [robot ID], [origin X], [origin Y]
A, 0, 0
B, 10, 0
C, 0, 10

# The assigned position in each time step. Format:
# [time step #], [robot ID], [visited position X], [visited position Y]
0, A, 1, 2
1, A, 1, 1
0, B, 4, 6
1, B, 4, 7
0, C, 1, 6
1, C, 6, 6
```

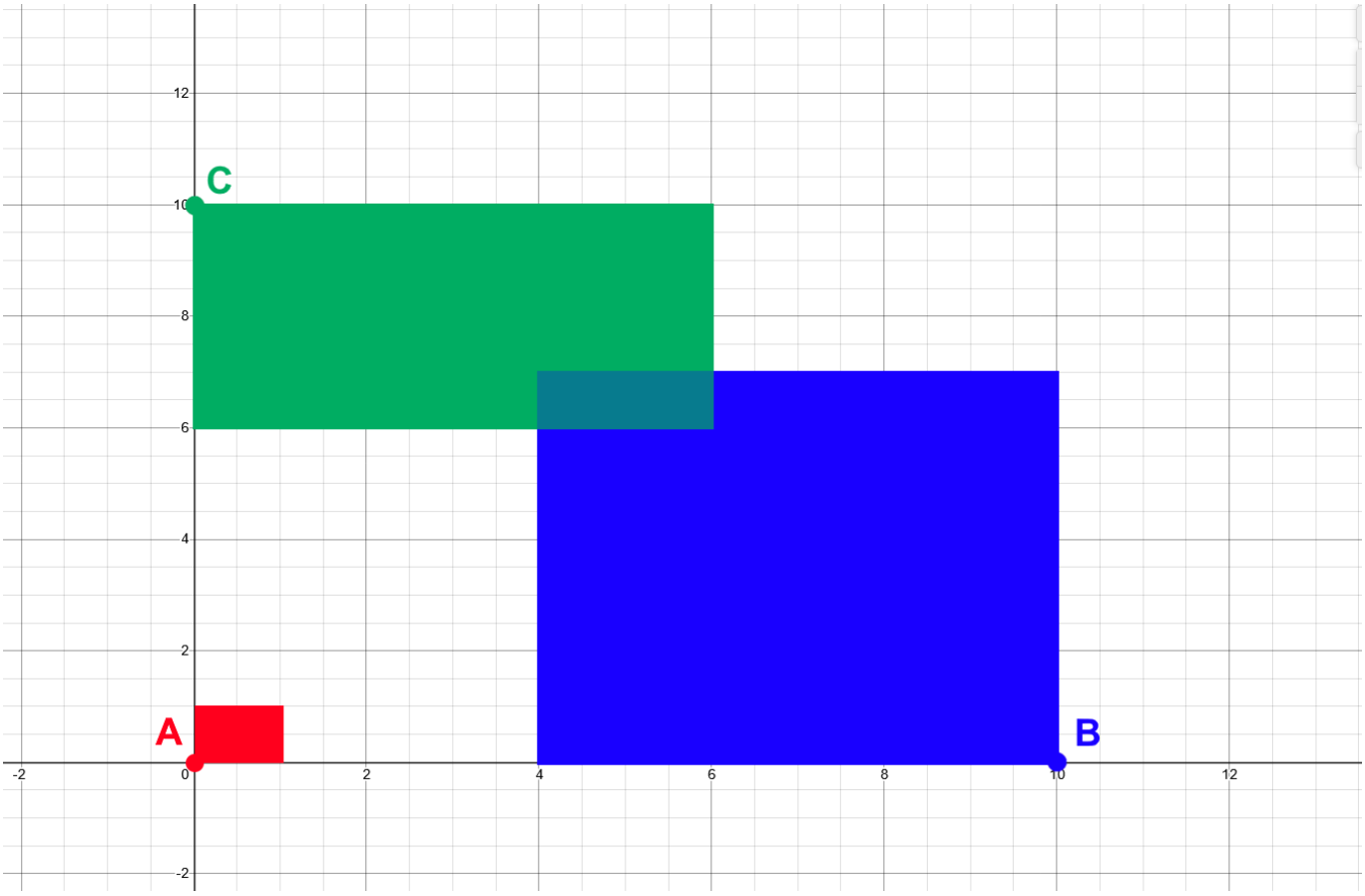
A schedule is valid if there are no collision between any of the robots for all time steps. A collision is defined as follows:

- Each robot occupies a rectangular area. The origin of the robot and the point that is being visited by the robot are the two corners of this rectangle.
- When the occupied areas of two robots overlap or touch in an edge or point, then there is a collision. See (Tip 1) for a formula that you can use to determine whether two rectangles overlap or touch.

In the sample schedule that is listed above, during time step 1, none of occupied areas of the robots A, B and C overlap (or touch), so there are no collisions. The occupied areas are indicated in the following image:



In the second time step, the area of robot B and C does overlap, so there is a collision in the second time step, causing the schedule to be invalid:



Requirements

The goal of this assignment is to implement a validation program, that checks whether a given schedule is valid. Read through the provided code and test cases before starting. The provided test cases will check if your code functions correctly. We will also manually check whether the code you submit is well written. Make sure your code is of high quality and easy to understand!

1. The input will always be valid, you do NOT have to implement any input validation. You can assume that the following statements hold:
    - There will always be at least 1 robot and at least 1 time step.
    - Every robot will have a unique ID.
    - The ID of a robot will never contain a comma.
    - For every time step, every robot will visit exactly 1 position.
    - There is no redundant whitespace. A comma is always followed by exactly 1 space.
    - Empty lines and lines with comments should be removed. There is already a function in the class `RobotScheduleValidationService` that does this for you.
2. You may not make changes to the file `Program.cs`
3. You may not change any of the provided test cases.
4. All provided test cases should pass.
5. Focus on creating code that is easy to understand, easy to test and easy to extend.
6. Create good abstractions for storing the parsed data.
7. Write code that others can easily understand!

Tips

1. To determine whether two rectangles overlap, you can use the following formula (you are also free to google a solution, or come up with a different formula by yourself):

```
Given rectangles A and B
The rectangles overlap, or touch on an edge/in a point, iff all of the following
expression holds:

A.Left <= B.Right
&& A.Right >= B.Left
&& A.Top >= B.Bottom
&& A.Bottom <= B.Top
```

2. Start by reading through the file `Service/RobotScheduleValidationService.cs`. The function `ValidateSchedule` is not fully implemented yet. Implement this function, such that it returns `true` if the schedule is valid and `false` if the schedule is invalid.
3. Run the tests in the project `NfiCodeAssignmentTest` and make sure that they pass before you hand in the assignment!
4. When you have finished the assignment:
    - Remove any unnecessary binaries (delete the `obj` and `bin` folders)
    - Create a single zip file that contains everything
    - Send the zip file by email