# Test Automation Answers

Please note, I refer to my drone object multiple times in these answers. The drone has multiple fields, including *x* and *y* which store its current position, as well as *bearing* which stores its bearing, which is then used to get the direction (North, South, East or West) when outputting to the user.

## Question 1

First, I would have to know what language the project is in as testing tools usually are language dependent. I would then narrow down my options to tools I have experience in. If I don’t have experience with any tools, I would choose a testing tool based on whether it can perform the functions I need. I would also prioritise tools that seem simpler to use, while still covering the required functionality. For example, a tool such as Jest would be a great fit for this application as it allows you to create and run basic unit tests, which would allow me to give it an input and ensure that the correct output is provided.

## Question 2

You can script paths you know will work as arrays of commands, or you can repeatedly randomly select a command from a list of commands to automate the movement of the drone. This should be written separately from the game code, so it doesn’t interfere with what is already working.

## Question 3

Because the movement of the drone is simple, it only moves forward in the direction it is facing, you can write unit tests to place it in a certain location, facing a certain direction, and the unit test can check to make sure that the drone’s x and y co-ordinates, as well as its bearing, are what they should be using assertations.

## Question 4

You can create an array or list of random commands and issue them one after the other. You can then use assertations to check that until a place command is issued, no other fields or values of the drone are changed. Further, you can return a Boolean from the command handling method which returns true if the command was successful and assert that this returns false until a place command is issued.

## Question 5

You can issue commands from a list which you know should send it out of bounds, and at the end assert that the drone’s current x and y co-ordinates are still within the boundary. On an empty surface, the list of commands could even just be the same command over and over until you know it would be out of bounds if the test was to fail.

## Question 6

My solution has obstructions that cannot be shot through or flown over. To check that it is not flying over an obstruction, you can assert that the drone’s current x and y positions are not within the list of obstructions (where the list of obstructions is made of x and y co-ordinates of obstructions). To check that the drone does not shoot through the obstructions, a similar method can be used, except instead of checking the block the drone is on, check the next two in front of it, as the drone fires two units ahead of its current position.

Further, I made a check that commands were valid, for instance a place command must have 2 parts, the command (PLACE) and arguments (X, Y and F). In order to be valid it has to have both parts, the second part must have 3 arguments, and the first 2 must be numeric. For the third argument to be valid, it must be one of the 4 specified directions.

Further, to check that the drone is turning, you can record the initial direction, as well as the direction after the command is issued, and ensure that the new position is either left or right of the initial position. I did this using bearing, as you can easily find the name of the direction in a dictionary which uses the bearing as a key, and you can easily rotate left and right by adding or subtracting 90 from the bearing.