1. List 3 reasons why it is good to write test cases.

- Ensures that program functions properly and fulfils design specification
- When specifications or circumstances are changed and the code needs to be modified, it allows us to check that the program's functionality has not been compromised by these changes
- Can pinpoint areas of code that need to be improved to fit design specification

2. We have not discussed mocks. Research and briefly explain the use of mocks in testing. List two advantages and two disadvantages of mocking. Where should mocks generally be used? (unit tests or E2E tests?)

Mocks should generally be used for unit testing, to test a unit that may have dependencies on other objects. Behaviour and interactions of the unit with the objects can be isolated and tested by replacing the dependencies with mocks, which simulate the behaviour of real objects.

Advantages:

- Allows testing for just one object without invoking its dependencies
- Allows testing if the real object is impractical to test

Disadvantages:

- Mocks test not only external but also internal behaviour of code, and may make refactoring harder because it links production code to test code
- Because mocking makes it harder to refactor, it may lead to the code design deteriorating over time, reducing design simplicity

3. Give a real life example of where insufficient code testing led to problems. Cite your resource appropriately. If you are unsure how to cite appropriately, use the Harvard referencing standard.

Software errors during Boeing Starliner's spacecraft's Orbital flight test stopped it from docking with the International space station, and may have caused problems during its planned re-entry into Earth's atmosphere. Segments of code were tested separately, however, no end-to-end tests were performed on the entire suite of code, so an incorrectly set mission elapsed timer caused the spacecraft to miss a planned engine firing, the bug undetected due to insufficient testing. (Clark, 2020)

Section 2 - Solver (7%): You may assume that the starting cell is on the top row and the ending cell is on the bottom row of the map. Qualitatively consider run-time and correctness.

1. What are the strengths and weaknesses of the BFS algorithm? When would you want to use BFS? (2%)

Strengths:

- BFS will never be trapped with unwanted nodes without solution
- Guaranteed to give optimal solution
- Finds closest goal quickly

Weaknesses:

- Takes a long time if solution is far away
- Uses a lot of memory, as each level of nodes are stored in memory

Use:

- Find shortest path
- When you know the solution node is not far from the root of the tree

2. What are the strengths and weaknesses of the DFS algorithm? When would you want to use DFS? (2%)

Strengths:

- Solution can be found quickly
- Finds the furthest solutions quickly
- Less memory used

Weaknesses:

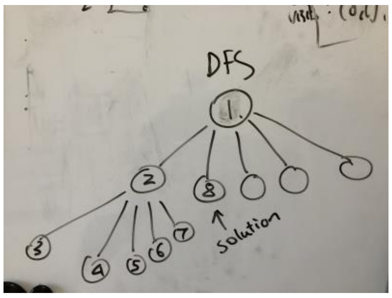
- Solution not guaranteed
- Can get trapped in searching useless paths without solution
- If multiple solutions are available, won't give the shortest solution
- Complexity depends on number of paths
- Cannot check duplicate nodes

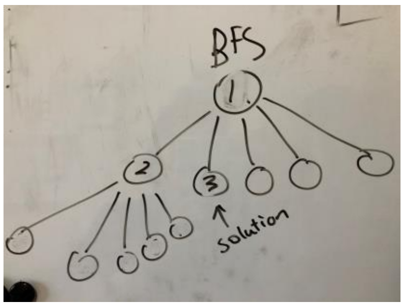
Use:

- Topological sorting
- Finding connected components

3. What happens if the ending cell is very close to the starting cell? Is DFS guaranteed to be faster than BFS? (1%)

DFS isn't guaranteed to be faster than BFS, because here the solution node is closer to the root of the tree. As seen in the diagram below, if the solution node is one vertex away from the root, BFS will find the solution faster (3 nodes searched compared to 8).





4. In normal BFS and DFS algorithms, there is usually a list of visited cells that help the algorithm perform faster by not re-visiting a cell. However, this game setup has a certain feature in it that does not allow you to use this. What feature of the game stops you from using a list of visited cells to improve run-time? Justify briefly by giving a simple example. (Hint: Think scenarios where you have to re-visit a cell to complete the game. What about you changed since the previous time you were there?)

The game can't use a list of visited cells to improve runtime because it certain cases, our game needs to revisit cells to complete the game- for example, in the 4x5 board below, in order to escape from being trapped after teleporting, the player must wait, **teleport to a previously visited cell** (1) to complete the game. So our DFS and BFS algorithms cannot just mark a cell as "visited" and prohibit itself from visiting it again.

```
**X**
**1  *
*1*  *
***Y*
```

Bibliography:

Bement, S. (2020). *To Mock or Not to Mock: Is That Even a Question? - SolutionsIQ*. SolutionsIQ. Retrieved 27 May 2020, from https://www.solutionsiq.com/resource/blog-post/to-mock-or-not-to-mock-is-that-even-a-question/.

Clark, S. (2020). *Boeing says thorough testing would have caught Starliner software problems – Spaceflight Now*. Spaceflightnow.com. Retrieved 27 May 2020, from https://spaceflightnow.com/2020/02/28/boeing-says-thorough-testing-would-have-caught-starliner-software-problems/.

Mock Testing. Devopedia. (2020). Retrieved 27 May 2020, from https://devopedia.org/mock-testing.

Quora. (2020). Retrieved 27 May 2020, from https://www.quora.com/What-are-the-advantages-of-using-BFS-over-DFS-or-using-DFS-over-BFS-What-are-the-applications-and-downsides-of-each.