Maze Report

Depth First Search (DFS):

A DFS of a maze involves following some path through the maze as far as we can go, until a dead end or previously visited location is met. When this occurs, the search backtracks to its most recent choice and tries a different path instead.

Breath First Search (BFS):

BFS proceeds differently: it visits the locations in order of their distance from the starting point of the search. First it visits all locations one step away, then it visits all locations that are two steps away, and so on, until the exit is found. Because of this, BFS has the nice property that it will naturally discover the shortest route through a maze.

Used Data Structures:

Queues: it's a data structure which put the data in a line every new input is put at the last of the queue and data which will be removed from it will be the first one put in it (FIFO).

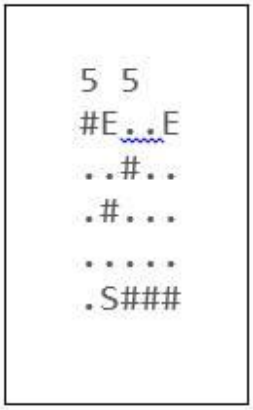
Stack: it's a data structure which put the data in a line and if you want to remove something from it you will remove the last data was put in the stack (LIFO).

Algorithms:

In DFS I used recursion with stack data structure which save the path in the stack will move in the maze in one way until it reaches the end of the path while moving we mark the place as visited then add this place in the stack after we got the path in the stack we add this stack information to an array then send it to the user.

In BFS I used loops with data structure queues that I use to save the path in, we use the same algorithm but we put the path in the queue then we mark the place as visited when we get the path, then we loop through it to put the path in the array.

Sample Run:



DFS path:

```
Point #1: 4, 1
Point #2: 3, 1
Point #3: 3, 2
Point #4: 2, 2
Point #5: 2, 3
Point #6: 1, 3
Point #7: 0, 3
Point #8: 0, 4
```

BFS path:

```
Point #1: 4, 1
Point #2: 3, 1
Point #3: 3, 0
Point #4: 2, 0
Point #5: 1, 0
Point #6: 1, 1
Point #7: 0,
```

Name: Muhammed Essam Khamis Ibrahim.

ID: 70.