Marwa Khalid (40155098)
Zafir Khalid (40152164)
COMP 352 – X
Sadegh Ghaderpanah
07 February 2021

ColonyExplorer – Iterative Pseudocode

```
static int ExploreAndLabelColony(2d array of characters called grid, integer row, integer
column, character label)
        Declare integer colonySize <- 0;
        Declare String coordinate <- integer row into String + "," + integer column into String;
        add the coordinate to the array called colonyCoordinates;
        Declare integer oldArraySize <- size of array colonyCoordinates;
        checkSurroundingElements(grid, integer row, integer column);

        while (size of coordinatesColony is not equal to oldArraySize) do
                Declare integer temp <- size of array colonyCoordinates;
                for integer l <- oldArraySize to less than temp do
                        Declare String coordinates <- colonyCoordinates element at index l
                        Declare String x <- coordinates from index 0 until index of "," – 1 (inclusive)
                        Declare String y <- coordinates from index of "," until the end (inclusive)
                        checkSurroundingElements(grid, x into integer, y into integer);
                end for
                oldArraySize <- temp;
        end while

        for integer l = 0 to less than the size of colonyCoordinates do
                Declare String coordinates <- colonyCoordinates element at index l
                Declare integer x <- coordinates from index 0 until index of "," – 1 (inclusive)
                Declare integer y <- coordinates from index of "," until the end (inclusive)
                grid at element x, y <- label;
        end for

        colonySize <- size of colonyCoordinates;
        clearing the array colonyCoordinates;
        return colonySize;
end method
```

ColonyExplorer – Iterative Analysis

The reason arrays were chosen over the other available structures is because it is time and memory efficient. Arrays keep data-type intact which allows easy access to any element. On the other hand, stack only accesses the top. Hence, in arrays regardless of the number of elements the time complexity for accessing an array is O(1). As well, arrays are memory efficient. It isn't possible to have extra memory in an array. Hence, a memory overflow or shortage will never occur.

The time complexity for ExploreAndLabelColony is O(n^2). When it comes to loops the worst case is that you have to traverse through the entire array resulting in a time complexity of O(n). If we examine the method we notice that we have a for loop nested in a while loop, the time complexity of the nested loops would be O(n^2). Afterwards, another for-loop is conducted which is equivalent to O(n). Therefore, we have the following calculation: O(n^2) + O(n) which is equal to O(n^2) since we only care about the domain term.

The memory complexity for ExploreAndLabelColony is O(m*n). Since the matrix is m in length and n in width the total number of elements would be m*n. This means it took m*n space to run the code.