Matthew Soicher
z5160737

Question 2

Firstly, we begin the problem by determining all possible positions that we can place our rooks so that they are **not** under attack of any bishop. A rook is under attack by a bishop if they are in any diagonal position to it.

For each bishop position (a_i, b_i), we will be unable to place rooks in cells (x_i, y_i) which may be either diagonally up left $(a_i - i, b_i - i)$, diagonally up right $(a_i - i, b_i + i)$, diagonally down left $(a_i + i, b_i - i)$ or diagonally down right $(a_i + i, b_i + i)$.

Note that i has no correlation to the subscript i in a_i, b_i

This is under the constraint of $0 < i < n$ and $1 \leq a_i \pm i, b_i \pm i \leq n$.

We can create a 2D Array to represent our chess board and provide a visual of where we can place our rooks. We initialize all cells with the number 1 (denoting that we can place a rook here). For given bishop positions we place a 0 in its corresponding cell (denoting we cannot place a rook here) for obvious reasons.

Then, we go through a for loop and populate our 2D array, placing 0's whereby rooks will not be able to go as by the rules stated above.

Pseudocode

We are given k bishops and their respective positions, the below is pseudocode to give the idea.

```
int createBoard ($bish_i(a_i, b_i),\ int\ n$)

    create int board[n][n] and initialise to 1
    set all bishop positions ($a_i$, $b_i$) in board [$a_i$][$b_i$] = 0

    for (int i = 0; i < n; i++) {

        for all bishops $bish_i(a_i, b_i)$

            if ($a_i - i \geq 1$ and $b_i - i \geq 1$)
                board [$a_i - i$][ $b_i - i$] = 0
            if ($a_i - i \geq 1$ and $b_i + i \leq n$)
                board [$a_i - i$][$b_i + i$] = 0
            if ($a_i + i \leq n$ and $b_i - i \geq 1$
                board [$a_i + i$][$b_i - i$] = 0
            if ($a_i + i \leq n$ and $b_i + i \geq 1$
                board [$a_i + i$][$b_i + i$] = 0
        end for
    end for

return board
```

Matthew Soicher
z5160737

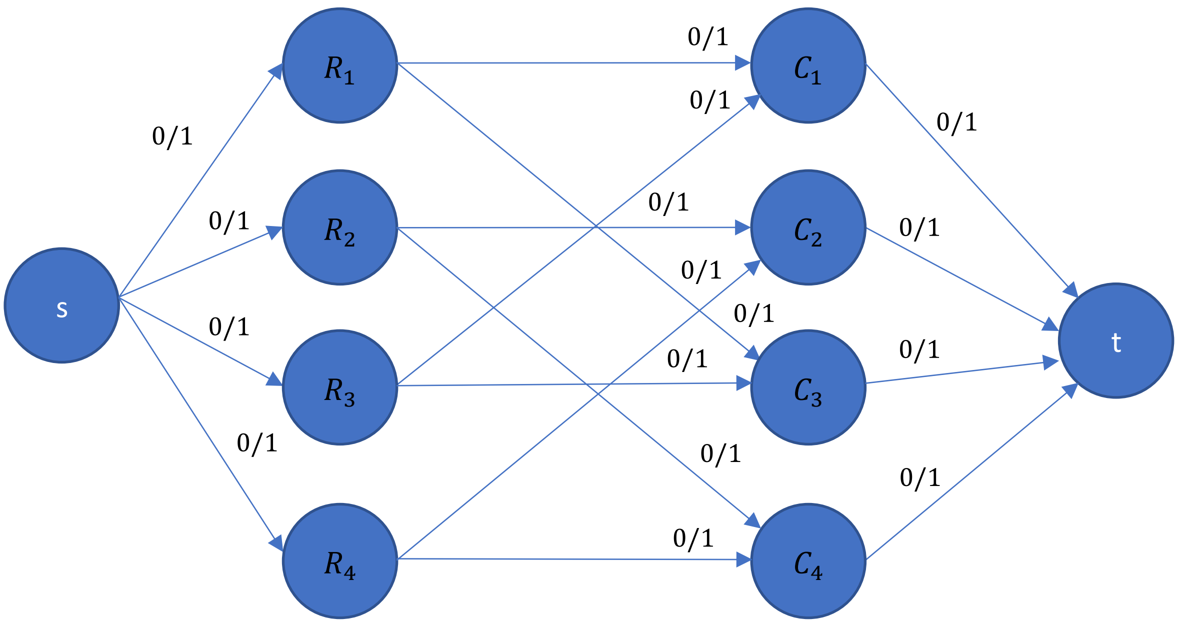
Now we know all possible positions whereby a rook can be placed (ignoring the constraint of rooks being unable to exist in the same row and column, for the moment).

We continue and construct a bipartite graph, whereby the vertices on the left denote the row index $1 \dots n$ and the vertices on the right denote the column index $1 \dots n$. If there exists a cell position $(i, j) = 1$ in our 2D array whereby a rook can be safe from a bishop as found above, we create a link from vertex i on the left to vertex j on the right. Below is an example to illustrate.

Say we have a 4x4 graph and our above code has created the 2D board array as so:

1	0	1	0
0	1	0	1
1	0	1	0
0	1	0	1

So, our bipartite graph will look like:



We denote a source vertex connecting to our row index vertices on the left, and our column index vertices on the right connect to a sink vertex. Due to the restriction that no two rooks are able to exist in the same row nor column, we denote the flow from one vertex (row) to the next vertex (column) with a capacity of 1.

Essentially, this is how one would approach the problem when solving it by inspection. If one chooses a rook to be placed at position (i, j) denoting a link from vertex i to vertex j, we will be unable to place any more rooks at either row i (the outgoing edge) and column j, the incoming edge. So, by restricting each connections quantity to 1, this method eliminates any possibility of placing two rooks in corresponding rows and columns.

And so, to maximise the number of rooks we can place on the board, we perform the max flow algorithm on this bipartite graph, which will fit our rule constraints and find an optimal solution.

Matthew Soicher
z5160737

This is the Ford-Fulkerson Algorithm, which runs in a time complexity of $\boldsymbol{O(E \times f)}$ where E = the number of edges in our graph and f = the maximum flow