Full Name: ____________________ Net ID: ____________________

CSO Honour Code
By turning in this exam for grading, I certify that I have produced the solution in accordance to the academic integrity policies of NYU Abu Dhabi as stated in `https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity`

Exam Instructions

1. You **must** follow all the x86-64 procedure call conventions.
2. You **must** show all steps of your work for full credit.
3. You **are** allowed to use a non-communicating calculator. You **cannot** use your phone, tablet, or laptop as a calculator. You **cannot** borrow a calculator from your colleagues.
4. You **are** allowed to have the reference sheet containing the summary of the x86-64 instructions, which I made available on Brightspace.
5. You **are** allowed one regular-size sheet of paper (front and back) on which you can write anything that you wish, as long as it is hand-written (i.e., not typed) by you (i.e., not by somebody else or a photocopy). You **are not** allowed to use any mechanical or electronic method of reproduction to create this sheet.

Question	Points	Score
Question 1	25	
Total:	25	

Question 1: (25 points)

N-Queen is the problem of placing N chess queens on an N×N chessboard, such that no two queens can attack each other. Figure 1 illustrates a solution for 4-Queen problem.

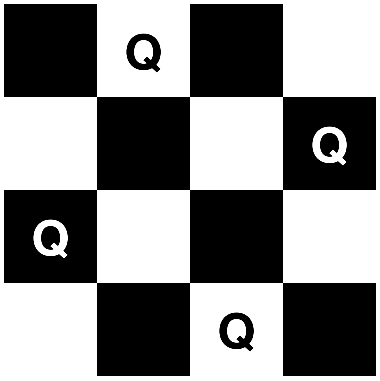

Figure 1: A solution for 4-Queen problem.

Luckily, there exist many programs in `C` that solves N-Queen. Your task is to write a x86-64 assembly function that must follow exactly the tasks performed in the `C` code for the function `solveNQueens` below (modified version of the code that is available at `https://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/`). You can assume that the x86-64 assembly code for the function `isSafe` is available to you.

```
// Returns 1 if a queen can be placed on board[row][col], 0 otherwise.
int isSafe(int** board, int N, int row, int col);

// A recursive function that solves the N Queens problem
int solveNQueens(int** board, int N, int col) {
    if (col >= N) return 1; // base case: all queens are placed, return 1

    // Within col, try placing this queen in all rows one by one
    for (int i = 0; i < N; i++) {
        if ( isSafe(board, N, i, col) == 1) {
            *(board + (i * N) + col) = 1;

            // recur to place rest of the queens
            if ( solveNQueens(board, N, col + 1) == 1 ) return 1;

            // Backtrack if no solution was found.
            *(board + (i * N) + col) = 0;
        }
    }

    return 0; // no solution found
}
```