ECE220 Computer Systems and Programming

Lab 7

1 After this week's lectures, you should be able to...

1. Explain how recursive functions solve their tasks and draw similarities to recurrence relations in math.
2. Identify the base case and the induction (recursive) case of a recursive function.
3. Describe briefly how activation records (also known as stack frames) are built-up and tear-down during a recursive call (details will be covered later in the course).
4. List the steps needed to solve a recursion with backtracking problem.

2 After today's lab, you should be able to...

1. Implement basic recursive backtracking algorithms.

3 Exercises

1. A helper function is a function that performs part of the computation for another function. It is used to improve code readability. The pseudo-code below solves a Sudoku game, and is provided on MP7's Wiki page. Read the algorithm and identify 1-2 places that could use a helper function. Provide the line number, function signature and explain the meaning of your arguments and return value. For example, line 10 could use a helper function:

```
// Checks the legality of filling cell (i, j) with val in sudoku.
// It returns 1 if val is valid and 0 otherwise.
int is_val_valid(const int val, const int i, const int j,
                 const int sudoku[9][9]);
```

```
1   bool solve_sudoku(int sudoku[9][9])
2   {
3     int i, j;
4     if (all cells are assigned by numbers) {
5       return true;
6     } else {
7       find a non-filled (0) cell (i, j)
8     }
9     for (int num = 1; num <= 9; num++) {
10      if (cell (i, j) can be filled with num) {
11        sudoku[i][j] <- num;
12        if (solve_sudoku(sudoku)) {
13          return true;
14        }
15        sudoku[i][j] <- non-filled (0);
16      }
17    }
18    return false;
19  }
```

Line 4

```
// return 1 if sudoku is complete, else return 0
int is_sudoku_complete(const int sudoku[9][9]) {
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            if (sudoku[i][j] == 0) {
                return 0;
            }
        }
    }
    return 1;
}
```

Line 7

```
// return (i, j) pair, a non-filled cell that can be filled with num
struct pair find_enpty_cell(const int sudoku[9][9]) {
    struct pair ij;
    ij.first = -1;
    ij.second = -1;
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            if (sudoku[i][j] == 0) {
                ij.first = i;
                ij.second = j;
                return ij;
            }
        }
    }
    return ij;
}
struct pair {
    int first;
    int second;
};
```