

1) (10 pts) DSN (Recursive Coding)

Consider writing a recursive method that counts the number of paths from a starting (x, y) location on the Cartesian plane to an ending (x, y) location. Let the starting location be (sx, sy) and the ending location be (ex, ey) , where all four coordinates are integers with $sx \leq ex$ and $sy \leq ey$, and for each step on a valid path, either 1 must get added to the current x coordinate or 1 must get added to the current y coordinate. In addition, some given locations are disallowed as intermediate locations on the path. **Complete the function shown below** to solve this task. The input to the function takes in sx, sy, ex, ey and a two dimensional integer array named *allowed*, such that *allowed[x][y]* is set to 1 if a path is allowed to go on coordinate (x, y) or set to 0 otherwise. It is guaranteed that (sx, sy) and (ex, ey) are coordinates which are both inbounds and an inbounds function is provided for you. It's not guaranteed that both (sx, sy) and (ex, ey) are valid locations to be on. In this case, the answer is 0.

```
#define N 10
int inbounds(int x, int y);

int numpaths(int sx, int sy, int ex, int ey, int allowed[][N]) {

    if (!allowed[sx][sy]) return 0;           // 1 pt

    if (sx > ex || sy > ey) return 0;           // 1 pt

    if (sx == ex && sy == ey) return 1;         // 1 pt

    int res = 0 ;                             // 1 pt

    if ( inbounds(sx+1, sy) )                  // 1 pt

        res += numpaths(sx+1, sy, ex, ey, allowed ); // 2 pts

    if ( inbounds(sx, sy+1) )                  // 1 pt

        res += numpaths(sx, sy+1, ex, ey, allowed ); // 2 pts

    return res;
}

int inbounds(int x, int y) {
    return x >= 0 && x < N && y >= 0 && y < N;
}
```

Grading Note: To earn 1 pt slots, answers must be perfect. On the two pt lines, award 2 pts if all 5 slots are correct, award 1 pt if at least 2 slots are correct, the order of the if statements doesn't matter but the inbounds check must correspond to the recursive call in its if statement.