**Computer Science I: Quiz 1 (Programs 1 - 3)**

**Last: _____ , First: _____**

1) (15 pts) A knight in chess can move in one of eight directions, all of which form an 'L' shape. If give each square on a board (x, y) coordinates, x for the row number and y for the column number, a knight on square (x, y) can move directly to

(x-2, y-1), (x-2, y+1), (x-1, y-2), (x-1, y+2), (x+1, y-2), (x+1, y+2), (x+2, y-1) and (x+2, y+1).

(a) Write two constant arrays, DX and DY, that encapsulate the relative movement of a knight. Please order your arrays in the manner that the eight ordered pairs above are listed (for ease of grading).

```
const int DX[] = {                                                    };

const int DY[] = {                                                    };
```

(b) Assume that a knight is on an infinite chess board, starting at (0, 0). Write a function that takes in an array of moves, where the moves are all integers in between 0 and 7, inclusive, representing the movement specified by the DX, DY arrays and prints out each location the knight visits from (0,0), NOT including the starting spot. As an example, if the move array stored [3, 2, 5, 5], then your function should print out

```
(-1, 2)
(-2, 0)
(-1, 2)
(0, 4)
```

Write the printMoves function as specified. (The input parameters are the array of moves, moves and length, the length of the array moves.)

```
void printMoves(int* moves, int length) {

    int x = 0, y = 0, i;









}
```

2) (10 pts) In the solution to the birthday problem, a function studentcmp is written. The function takes in pointers to two struct students and returns a negative integer if the first one comes before the second one, 0 if they are identical and a positive integer if the first one comes after the second one. Why is it better to write a function like this than embedding the logic of the function in the merge or partition functions?

3) (15 pts) Consider a different prompt for the mastermind assignment that takes in the same identical input:

Instead of outputting the number of possible color combinations that are consistent with the current transcript of the game, output the ***first*** possible color combination (in lexicographical ordering) that is consistent with the current transcript of the game.

One way to edit the currently posted solution to accomplish this task would be to rewrite the solve function. The solve function for the assignment takes in the combo array and an integer k, and returns the number of possible solutions where the first k slots of combo are fixed. In the edited version, this function would return null if there were no solutions under the given constraints and return an int* storing the appropriate color combination otherwise. Note: lexicographical ordering is simply the ordering the odometer code shown in class naturally does (0,0,0,0), (0,0,0,1),…(0,0,0,5), (0,0,1,0), etc. if there were 6 possible colors.

Here is the solve function posted in the solution:

```
int solve(int* combo, int k) {
    if (k == numSlots) return eval(combo);

    int i, sum = 0;
    for (i=0; i<numColors; i++) {
        combo[k] = i;
        sum += solve(combo, k+1);
    }
    return sum;
}
```

Recall that currently, eval returns 1 if combo is consistent with the guesses and feedback and 0 otherwise. Rewrite this function to do the alternative task at hand. Some of the edited function has been given to you.

```
int* solve(int* combo, int k) {

    int i;

    if (k == numSlots) {
        int res = eval(combo);
        if (res == 0)
            return ________ ;
        else {
            int* arr = _____________________________________________ ;

            ___________________________________

                ________________________________ ;

            return arr;
        }

    }

    for (i=0; i<numColors; i++) {
        combo[k] = i;
        int* array = solve(combo, k+1);

        if ( ___________________ )

            return ____________ ;
    }
    return __________ ;
}
```