COP 3502 Study Group Sheet: Recursion

Directions: Work together as a group to try to solve these problems. Talk through issues and see if you can convince yourselves of the right path to move forward. In groups with a TA/ULA, towards the end of the session some of the solutions will be covered. At the end of the week, the solutions will be posted for everyone.

Each of the following questions asks you to write a recursive function.

1) The code below returns the nth Harmonic number. (Note: n must be positive.) Rewrite the function recursively.

```
double harmonic(int n) {
    double res = 0;
    for (int i=1; i<=n; i++)
        res += 1.0/i;
    return resl
}
```

Rewrite this method ***recursively***:

```
double harmonic(int n);
```

2) Write a *recursive* function returns the sum of the digits of its input parameter n. You may assume that n is non-negative. For example, productDigits(274) should return 56, since 2*7*4 = 56.

```
int productDigits(int n);
```

3) Without running the function below, determine the output of the function call doit(4):

```
void doit(int n) {
    if (n>0) {
        doit(n-1);
        printf("%d ", n);
        doit(n-1);
    }
}
```

What is this function similar to, in structure?

4) The function below is an attempt at a recursive binary search of a sorted array. Why is this function no faster than a basic linear search through the array?

```
int search(int numbers[], int low, int high, int value) {

    if (low > high) return 0; // Not found.

    int mid = (low+high)/2;

    if (numbers[mid] == value) return 1; // Found.

    return search(numbers, low, mid-1) ||
           search(numbers, mid+1, high);
}
```

5) Imagine being a particle starting at the coordinates (x1, y1) in the Cartesian plane, moving to (x2, y2), where x1 ≤ x2 and y1≤ y2, and at each step you could either add 1 to your x coordinate or add 1 to your y coordinate. Write a recursive function to calculate the number of different ways to make the journey. (No need to code a base case for when it's not possible, ie when x1 > x2 or y1 > y2.)

```
int numWays(int x1, int y1, int x2, int y2);
```