Handout 19

Practice with functions

19.1 Quiz!

Write a function

```
void arrayInit(double array[], int size)
```

that takes in an array of size `size` and fills the array with zeroes.

19.2 Homework: `random-walkers.c`

Description

This program will send two beings on a random walk. Each random walk starts at $(x, y) = (0, 0$ and consists of `steps` number of steps of size `stepsize`. (If `steps=5` and `stepsize = 2.5` then each random walk has 5 steps where the x location changes randomly by up to distance ± 2.5 and the y location changes randomly by up to distance ± 2.5.) At the end of the random walk, the program computes the distance between the two beings. The program then does this a large number of times, computing the average final distance.

In this program, the location of the two beings are given by arrays

```
double locA[2];
double locB[2]
```

where `locA[0]` is the x coordinate of the first being, `locA[1]` is the y coordinate, etc.

Functions

The program must use the following functions

```
// function to complete a single random step
void randStep(double location[], double stepsize)
```

```
// function to do one random walk by taking multiple steps
void randWalk(double location[], int numberSteps, int stepsize)
```

```
// function to compute distance
double computeDistance(double location1[], double location2[])
```

These functions should be wrapped in to a single function

```
double trial(int numberSteps, double stepsize)
{
  double locA[2];
  double locB[2];
  arrayInit(locA,2);
  arrayInit(locB,2);
  randWalk(locA, numberSteps, stepsize);
  randWalk(locB, numberSteps, stepsize);
  return computeDistance(locA,locB);
}
```

I suggest that you start by running `trial` once, and then put it in a loop.

The assignment

Your program should ask for the following inputs *in this order*:

- how many steps each walk should have (`int`)
- how big each step can be (`double`)
- how many different walks we want to run

Your program should give out the average final distance of the walks that it simulated. Finally, you code needs to have good comments!!

19.3 Lab: Prime numbers revisited

Write program for which

- the input is a positive integer and
- the output is the list of all primes up to (and including, if appropriate) the input. The output should be delivered all on the same line.

Do this using the following method:

- Create an array with the number of slots determined by the input.
- Fill the array with the numbers `1,2,3,..., input`
- Replace each entry that is not prime with a zero using the following method:
  - replace the leading `1` with zero
  - for all numbers larger than the `2`, replace them with zero if they are divisible by `2`
  - repeat previous step for `3`
  - keep doing this until you need to stop.

You code should

- use user-defined functions to get the input, perform the various tasks, and deliver the output; and
- have nice comments.