Handout 12

Nested loops

12.1 Loops inside loops

Suppose we want to add all of the following numbers

```
1
1 2
1 2 3
1 2 3 4
...
1 2 3 4 ... 10
```

We can build a loop to compute the sum in row `k`... and put that loop inside yet another loop that adds over all rows.

12.2 Practice

Write code where the input is an integer n and the output is the result of

$$\left(1\right) + \left(1 + \frac{1}{2}\right) + \left(1 + \frac{1}{2} + \frac{1}{3}\right) + \cdots + \left(1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n}\right)$$

12.3 More practice

Consider the following game: A coin is randomly flipped 10 times and the points earned is the number of "heads" that appear. Thus the possible number of points ranges from 0 to 10. Write code that plays this game 100 times. Store the results of how many points are earned in an array. Then (using a loop!) print out how many times each point value was earned. The result should look like this:

```
0 points: ??? times
1 points: ??? times
2 points: ??? times
```

```
...
10 points: ??? times
```

12.4 Homework: `birthdays.c`

Start with the "birthdays" code discussed in class last week. Put this code inside a loop that runs the procedure one million times. What percent of the time is there a shared birthday?

There are several times in the birthday code that we need to convert between `double` and `int`. Here is a good way to do the conversions.

- Technically, the output of the `floor()` function is a `double`. To convert the output to type `int` use the code

```
day = (int)floor(????);
```

- Suppose we have two integers `count` and `total`, and we want to divide in order to compute a percent. The best way to do this is to first convert `count` to type `double` and then divide. Here is the code that does this:

```
percent = ( (double)count )/total;
```

Additional challenge: Use your code to determine the smallest class size needed to have a greater than 50% chance of two people sharing the same birthday.

12.5 Homework: `many-random-walks.c`

Start with the random-walk code from Exam 1. When choosing a random number between -1 and 1, be sure that you are using the procedure

```
drand48()*2-1
```

Put your random walk code inside a loop that performs one million random walks. As you do these walks, keep track of the final distance from $(0, 0)$ that the motion achieves. Print out a table that reports which percent of walks has the final distance in the range 0-1, 1-2, 2-3, etc. Present your data in the following format:

```
Distance     Percent
-------------------------
0 - 1        ????
1 - 2        ????
2 - 3        ????
3 - 4        ????
...
9 - 10       ????
```