Extra Credit 4

Due: June 8, 2023 **Points:** 50

The *birthday problem* asks how many people must be in a room so that the probability of two of them having the same birthday is 0.5. This problem has you explore it by simulation. Basically, you will create a series of lists of random numbers of length $n = 2, \ldots$, and look for duplicates. You will do this 5000 times for each length. For each length, count the number of lists with at least 1 duplicate number; then divide that number by 5000. That is the (simulated) probability that a list of *n* generated numbers has at least one duplicate. As the random numbers you generate are between 1 and 366 (each one corresponding to a day of the year), this simulates the birthday problem.

Now, breathe deeply and calm down. We will do this in steps!

1. First, detecting duplicates. Write a function called `hasduplicates(`*`A`*`, textitnA)` that takes an array *`A`* and returns 1 if it contains a duplicate element, and 0 if it does not. For example:

   ```
   int A[] = { 1, 2, 3, 4, 5, 5, 2 };
   int nA = 7;
   /* . . . */
   x = hasduplicates(A, nA)
   ```

   After this, `x` is 1.

   ```
   int A[] = { 1, 2, 3, 4, 5, 6, -1 };
   int nA = 7;
   /* . . . */
   x = hasduplicates(A, nA)
   ```

   After this, `x` is 0.

2. Now, deal with one set of birthdays. Write a function called `onetest(`*`count`*`)` that generates a list of *`count`* random integers between 1 and 366 inclusive, and returns 1 if it contains a duplicate element, and 0 it does not. Please use the function `hasduplicates(`*`A`*`, `*`nA`*`)` to test for duplicates.

3. Now for the probability for *count* people. Write a function `probab(`*`count`*`, `*`num`*`)` that runs *num* tests of *count* people, and counts the number of tests with duplicates. It returns the fraction of the tests with duplicates; that is, the number of duplicates divided by *num*.

   To generate the random numbers, use *`drand48`*, which generates a double between 0.0 and 1.0 (it will never genertate 1.0, though). Before your first call, initialize the pseudorandom number generator by using *`srand`*. Here's an example:

   ```
   double d;

   srand(time(NULL));
   /* . . . */
   d = drand48();
   ```

4. Now for the demonstration. Start with 2 people, and begin adding people until the probability of that many people having two people with a birthday in common is over 0.9. (In other words, start with a list of 2 elements, and increase the number of elements in the list until the simulation shows a probability of 0.9 that a number in the list is duplicated.) Print each probability; your output should look like this:

   ```
   For  2 people, the probability of 2 birthdays is 0.00220
   For  3 people, the probability of 2 birthdays is 0.00880
   For  4 people, the probability of 2 birthdays is 0.01680
   ```

```
For  5 people, the probability of 2 birthdays is 0.02940
For  6 people, the probability of 2 birthdays is 0.03940
For  7 people, the probability of 2 birthdays is 0.05900
For  8 people, the probability of 2 birthdays is 0.06840
For  9 people, the probability of 2 birthdays is 0.09700
For 10 people, the probability of 2 birthdays is 0.12360
```

How many people are needed so that the probability of two of them with a birthday in common is over 0.9? How many are needed such that the probability of two of them having the same birthday is at least 0.5? Put these answers into a comment at the head of the file.

Hint: Don't be surprised if your probabilities are slightly different than the ones shown in the sample output. As randomness is involved, it is very unlikely your numbers will match the ones shown here.

Call your program "bday.c". Submit it through Canvas; do not submit it through Gradescope.