CSC112 Fundamentals of Computer Science Spring 2016
Lab 5 – Happy Birthday

You may have heard of a popular party trivia question: "Who has the same birthday ?" Here "birthday" means which day of the year, not including the exact year of their birth. If you are having a large gathering, i.e., with more than 366 people,[1] it is obviously a certainty that there must be at least two people in the group with with the same birthday.[2]

But, what if you have fewer people at your big bash, say 20 or 30 ? If you ask most people, they would guess that it is unlikely that two people out of 30 have the same birthday. They would be wrong. It takes surprisingly few people in a group to have a high probability that two or more people have the same birthday. The goal of this lab is to estimate the probabilities that in a group of N people, two or more people will have the same birthday, where $2 \leq N \leq 50$. We will use simulation to estimate the probabilities.
Your program *must* be written in an object-oriented style using C++ classes.

Input: None.

Output: A table of the number of people (from 2 to 50) and the estimated probability that a group of that size has at least two people with the same birthday.

Implementation Guidelines:

- To be accurate, we need to consider the possibility that some people may be born on leap year's day. A simple implementation is to consider a 4 year period[3]. There are: $365 + 365 + 365 + 366 = 1461$ days in a four year period. If we generate a random number x in the inclusive range 0 to 1460, and if we designate the outcome $x = 1460$ as "born on leap year's day", then there will be the correct probability (1/1461) that our simulated person was born on leap year's day.

  The remaining $4 \times 365 = 1460$ outcomes, numbered 0 to 1459, must be aggregated in groups of 4 to simulate the 4 times that a person with a non-leap birthday will celebrate during our hypothetical 4-year period. A simple way to aggregate the remaining outcomes into non-leap days is to compute x modulo 365, giving you a number in the inclusive range 0 to 364.

  Suggestion: Use the integers 0 through 365 to represent day (of the year) on which a simulated person is born. The range 0 to 364 can represent non-leap days, and 365 indicates a birth on a leap year's day.

- To simulate a group size N, generate N birthdays at random as described above. If any two simulated persons have the same birthday, then count this trial as a "success".

[1] It's 366 not 365 because some people may be born on February 29
[2] This logical proof technique is often called "pigeon hole principle".
[3] For our purposes, we will ignore century and millennia years.

- For a fixed group size N, repeat the simulated trial 100,000 times. Suppose you get r outcomes that are considered 'success". The estimate of the desired probability is $p = r/100000$.
- Repeat the probability estimates for every N in the range $2 \leq N \leq 50$, and print a table.

Coding Guidelines:

1. Your implementation must include at least one class; The number of classes you use depends on your approach to the problem, but you probably do not need more than one class.
2. Data members must be private. Provide access to private data via public functions only as needed.
3. Your class must provide at least two public methods:
   - `void est_probs()`
   - `void show_probs()`

   Member function `est_probs()` computes the table of probabilities described above. The member function `void show_probs()` prints the table.
4. It is less-than-desirable program design to do the entire implementation in function `est_probs()`. You should break the computation in to simple-sized private functions. One possible organization is:
   (a) Function `get_birthday` generates a random birthday, including the details of leap year's day described above.
   (b) Function `one_group` simulates a group of n people, and returns an indication whether that group has two or more simulated people with the same birthday,
   (c) Function `one_estimate` repeats the simulation of n people m times.
   (d) Function `est_probs` uses $m = 100000$ and gets a probability estimate for groups size $2, 3, 4, ..., 50$.
   (e) Use private arrays and data as needed to support the public and private member functions.
5. The main program should be very simple. If the class is named `simulate`, then the following main program is sufficient.

```
int main()
{
   simulate B ;

   B.est_probs() ;
   B.show_probs() ;
}
```

Running Your Program

```
gottlieb% a.out > probs
```

We will use a program named "octave" to draw a plot for us, showing the relationship between group size and the probability that at least two people have the same birthday. Save your plot to a file name "probs.png".

The octave commands you need are shown below:

```
octave:1> load probs
octave:2> npeople = probs(:,1) ;
octave:3> pr = probs(:,2) ;
octave:4> plot(npeople, pr)
octave:5> title("Two (or more) People Have the Same BIrthday") ;
octave:6> xlabel("Number of People") ;
octave:7> ylabel("Probability") ;
octave:8> print("probs.png")
```

Turn In;

Question: What's the minimum number of people in a group to have at least a 50% chance that two (or more) people have the same birthday ? Are you surprised by the answer ?

Save all your work in a directory named "Lab5". Change to your home directory (the parent directory of "Lab5"), and create a file named "lab5.tar" using the command:

```
tar cf lab5.tar Lab5
```

Use `sftp` to upload the file "lab5.tar" to your account on telesto.

It's My Birthday Too, Yeah.