CSCI 135 Software Design and Analysis, C++
Lab 5 Solution

Saad Mneimneh
Hunter College of CUNY

Lab A: The birthday paradox
One would think that in a small set of randomly chosen people, the probability that some pair of them have the same birthday is small. In this exercise, we will simulate this setting to find out how many people are needed for this probability to reach certain values. Surprisingly, this number will be much smaller than 367 (there are 366 possible birthdays).

We will assume that n, the number of people, is a constant integer of your choice, that you get to change whenever you want.

```
#include <cstdlib>

int main() {
  srand(time(0)); //initializes the random number generator

  const int n=...;

  . . .

}
```

(a) Declare an array of int of size n and initialize the array randomly with numbers in the range $[1, 366]$.

(b) Find if there is some duplicate.

(c) Repeat (a) and (b) 10000 times (using a loop) to count how many times some duplicate is found, and divide by the number of trials 10000 to find the probability.

(d) Adjust the value of n (for now manually) to make this probability around 50%, 90%, and 99%. What are those values of n?

(d) Convert the program you have done into a function that accepts n as a parameter and returns the probability associated with n. Set the size of your array to 367 regardless of n, but use it as if it is an array of size n. In main, make a loop that goes through all n from 1 to 367, and find the probability for that n.

Solution parts (a), (b), and (c):

```
int main() {
  srand(time(0));
  const int n=...; //whatever you want
  int a[n];
  int count=0;
  for (int trial=1; trial<=1000; trial=trial+1) {
    //init
    bool duplicate=false;
    for (int i=0; i<n; i=i+1)
      a[i]=rand()%366+1;

    //check for dups
    for (int i=0; i<n; i=i+1) {
      for (int j=i+1; j<n; j=j+1)
        if (a[i]==a[j]) {
          duplicate=true;
          break;  //not really needed, but for efficiency
        }
      if (duplicate)
        break;  //not really needed, but for efficiency
    }
    if (duplicate)
      count=count+1;  //a duplicate has been found
  }
  cout<<count/1000.0;
}
```

n=23, probability around 50%
n=41, probability around 90%
n=56, probability around 99%

Solution part (d):

```
float birthday(int n) {
  int a[367]; //don't use n>367; otherwise, use dynamic mem allocation
  int count=0;
  for (int trial=1; trial<=1000; trial=trial+1) {
    //init
    bool duplicate=false;
    for (int i=0; i<n; i=i+1)
      a[i]=rand()%366+1;

    //check for dups
    for (int i=0; i<n; i=i+1) {
      for (int j=i+1; j<n; j=j+1)
        if (a[i]==a[j]) {
          duplicate=true;
          break;  //not really needed, but for efficiency
        }
      if (duplicate)
        break;  //not really needed, but for efficiency
    }
    if (duplicate)
      count=count+1;  //a duplicate has been found
  }
  return count/1000.0;
}

int main() {
  srand(time(0));
  for (int n=1; n<=367; n=n+1)
    cout<<n<<' '<<birthday(n)<<'\n';
}
```

With dynamic memory allocation:

```
float birthday(int n) {
  int * a=new int[n];  //allocate memory
  int count=0;
  for (int trial=1; trial<=1000; trial=trial+1) {
    //init
    bool duplicate=false;
    for (int i=0; i<n; i=i+1)
      a[i]=rand()%366+1;

    //check for dups
    for (int i=0; i<n; i=i+1) {
      for (int j=i+1; j<n; j=j+1)
        if (a[i]==a[j]) {
          duplicate=true;
          break;  //not really needed, but for efficiency
        }
      if (duplicate)
        break;  //not really needed, but for efficiency
    }
    if (duplicate)
      count=count+1;  //a duplicate has been found
  }
  delete[] a;  //free memory
  return count/1000.0;
}
```