PRINCETON UNIVERSITY

School of Engineering and Applied Science
Department of Computer Science

COURSE	Computer Science : Programming with a Purpose
LESSON	Arrays
INSTRUCTOR	Ph.D Robert Sedgewick

MODULE 03 - PROGRAMMING ASSIGNMENT

The purpose of this assignment is to give you practice writing programs with loops and conditionals.

1. **Discrete Distribution**
   Write a program **DiscreteDistribution.java** that takes an integer command-line argument **m**, followed by **a** sequence of positive integer command-line arguments $a_1, a_2, \ldots, a_n$, and prints m random indices (separated by whitespace), choosing each index **i** with probability proportional to a_i.

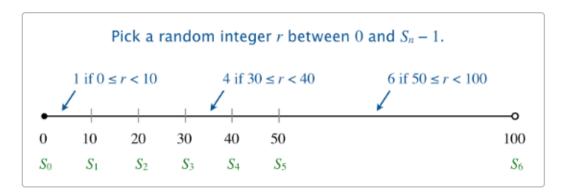
To generate a random index **i** with probability proportional to a_i **:**

- Define the cumulative sums $S_i = a_1 + a_2 + \ldots + a_n$
- Pick a random integer **r** uniformly between **0** and $S_n - 1$**.**
- Find the unique index **i** between **1** and **n** such that $S_{i-1} \leq r \leq S_i$.

Geometrically, this subdivides the interval $[0, S_n)$ into n subintervals $[S_{i-1}, S_i)$ with the length of subinterval i proportional to a_i. For example, if the discrete distribution is defined by

$(a_1, a_2, a_3, a_4, a_5, a_6) = (10, 10, 10, 10, 10, 50)$, *then the cumulative sums are* $(S_1, S_2, S_3, S_4, S_5, S_6) = (10, 20, 30, 40, 50, 100)$ *and the following diagram illustrates the 6 subintervals:*

In probability theory, this is known as sampling from a discrete distribution.

2. **Thue-Morse weave**

   Write a program ThueMorse.java *that takes an integer command-line argument n and prints an n-by-n pattern like the ones below. Include two space characters between each + and - character.*

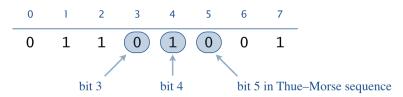*The Thue–Morse sequence is an infinite sequence of 0s and 1s that is constructed by starting with 0 and successively appending the bitwise negation (interchange 0s and 1s) of the existing sequence. Here are the first few steps of this construction:*
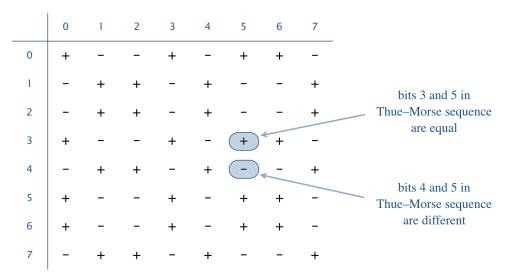*0*
01
0110
01101001
0110100110010110

To visualize the Thue–Morse sequence, create an n-by-n pattern by printing a **+** *(plus sign) in row* ***i*** *and column* ***j*** *if bits* ***i*** *and* ***j*** *in the sequence are equal, and a* **–** *(minus sign) if they are different.*

Thue–Morse sequence (n = 8)

0	1	2	3	4	5	6	7
0	1	1	0	1	0	0	1

bit 3   bit 4   bit 5 in Thue–Morse sequence

Thue–Morse weave (n = 8)

	0	1	2	3	4	5	6	7
0	+	–	–	+	–	+	+	–
1	–	+	+	–	+	–	–	+
2	–	+	+	–	+	–	–	+
3	+	–	–	+	–	+	+	–
4	–	+	+	–	+	–	–	+
5	+	–	–	+	–	+	+	–
6	+	–	–	+	–	+	+	–
7	–	+	+	–	+	–	–	+

bits 3 and 5 in Thue–Morse sequence are equal

bits 4 and 5 in Thue–Morse sequence are different

Note: you may assume that n is a positive integer (but it need not be a power of 2).

The Thue–Morse sequence has many fascinating properties and arises in graphic design and in music composition.

3. ***Birthday problem***

    Suppose that people enter a room one at a time. How people must enter until two share a birthday? Counterintuitively, after 23 people enter the room, there is approximately a 50–50 chance that two share a birthday. This phenomenon is known as the birthday problem or birthday paradox.

    Write a program `Birthday.java` that takes two integer command-line arguments n and trials and performs the following experiment, trials times:
      ● *Choose a birthday for the next person, uniformly at random between 0 and*
      ● *n−1.*
      ● *Have that person enter the room.*
      ● *If that person shares a birthday with someone else in the room, stop; otherwise repeat.*

    In each experiment, count the number of people that enter the room. Print a table that summarizes the results (the count i, the number of times that exactly i people enter the room, and the fraction of times that i or fewer people enter the room) for each possible value of i from 1 until the fraction reaches (or exceeds) 50%.

i	count	fraction	
1	0	0	
2	2710	0.002710	
3	5547	0.008257	
4	8105	0.016362	
5	10776	0.027138	among 1 million experiments, fraction in which first duplicate birthday happens before 6th person enters (2,710 + 5,547 + 8,105 + 10,776) / 1,000,000
6	13413	0.040551	
7	15782	0.056333	
8	17816	0.074149	
9	20283	0.094432	
10	22297	0.116729	among 1 million experiments, number in which first duplicate birthday happens when 5th person enters
11	24105	0.140834	
12	26013	0.166847	
13	27247	0.194094	
14	28405	0.222499	
15	29873	0.252372	
16	30447	0.282819	
17	31445	0.314264	
18	31837	0.346101	
19	32559	0.37866	
20	32244	0.410904	
21	32357	0.443261	
22	32020	0.475281	
23	31667	0.506948	stop (fraction exceeds 1/2)

```
~/Desktop/arrays> java Birthday 365 1000000
1       0       0.0
2       2710    0.00271
3       5547    0.008257
4       8105    0.016362
5       10776   0.027138
6       13413   0.040551
7       15782   0.056333
8       17816   0.074149
9       20283   0.094432
10      22297   0.116729
11      24105   0.140834
12      26013   0.166847
13      27247   0.194094
14      28405   0.222499
15      29873   0.252372
16      30447   0.282819
17      31445   0.314264
18      31837   0.346101
19      32559   0.37866
20      32244   0.410904
21      32357   0.443261
22      32020   0.475281
23      31667   0.506948

~/Desktop/arrays> java Birthday 31 1000000
1       0       0.0
2       32270   0.03227
3       62580   0.09485
4       87582   0.182432
5       105596  0.288028
6       114427  0.402455
7       115494  0.517949
```

The birthday problem arises in many computer science applications, including hashing, cryptographic attacks, and testing random number generators.

_____

4. Minesweeper

Minesweeper is a 1960s era video game played on an m-by-n grid of cells. The goal is to deduce which cells contain hidden mines using clues about the number of mines in neighboring cells. Write a program `Minesweeper.java` *that takes three integer command-line arguments* **m, n,** *and* **k** *and prints an* **m**-by-**n** *grid of cells with* **k** *mines, using asterisks for mines and integers for the neighboring mine counts (with two space characters between each cell). To do so,*

- *Generate an* **m**-by-**n** *grid of cells, with exactly* **k** *of the mn cells containing mines, uniformly at random.*
- *For each cell not containing a mine, count the number of neighboring mines (above, below, left, right, or diagonal).*

```
~/Desktop/arrays> java Minesweeper 9 9 10
0  1  *  1  0  0  0  1  *
1  3  2  2  0  0  0  1  1
*  2  *  1  0  0  1  1  1
1  2  2  2  1  0  1  *  1
0  1  2  *  1  0  1  1  1
1  2  *  3  3  1  1  0  0
1  *  3  *  2  *  1  0  0
1  1  2  1  2  1  1  0  0
0  0  0  0  0  0  0  0  0
```

Submission. Submit a `.zip` file containing `DiscreteDistribution.java`, `ThueMorse.java`, `Birthday.java`, and `Minesweeper.java`. You may not call library functions except those in the `java.lang` (such as `Integer.parseInt()` and `Math.sqrt()`). Use only Java features that have already been introduced in the course (e.g., loops and arrays, but not functions).

_____