AN OBJECT-ORIENTED RANDOM-NUMBER PACKAGE IN JAVA WITH MANY LONG STREAMS AND SUBSTREAMS

This class implements the `RngStream` class, with a few additional tools. The backbone generator is the combined multiple recursive generator (CMRG) `Mrg32k3a` proposed in (L'Ecuyer 1999), implemented in 64-bit floating-point arithmetic. This backbone generator has period length $\rho \approx 2^{191}$. The values of V, W, and Z are 2^{51}, 2^{76}, and 2^{127}, respectively. The seed of the RNG, and the state of a stream at any given step, are 6-dimensional vectors of 32-bit integers. The default initial seed of the package is $(12345, 12345, 12345, 12345, 12345, 12345)$.

```
public class RngStream {
```

`public RngStream ()`
Constructs a new stream, initializes its seed I_g, sets B_g and C_g equal to I_g. The seed I_g is equal to the initial seed of the package given by `setPackageSeed` if this is the first created stream, otherwise it is Z steps ahead of the seed of the most recently created stream. Also sets its antithetic and increased precision switches to `false`.

`public RngStream (String name)`
Constructs a new stream with identifier `name` (used when printing the full state of the stream).

`public static boolean setPackageSeed (long seed[])`
Sets the initial seed for the class `RngStream` to the six integers in the vector `seed[0..5]`. This will be the seed (initial state) of the first stream. If this procedure is not called, the default initial seed is $(12345, 12345, 12345, 12345, 12345, 12345)$. If it is called, the first 3 values of the seed must all be less than $m_1 = 4294967087$, and not all 0; and the last 3 values must all be less than $m_2 = 4294944443$, and not all 0. Returns `false` for invalid seeds, and `true` otherwise.

`public void resetStartStream ()`
Reinitializes the stream to its initial state: C_g and B_g are set to I_g.

`public void resetStartSubstream ()`
Reinitializes the stream to the beginning of its current substream: C_g is set to B_g.

`public void resetNextSubstream ()`
Reinitializes the stream to the beginning of its next substream: N_g is computed, and C_g and B_g are set to N_g.

`public void setAntithetic (boolean a)`
After this procedure is called with `a = true`, the stream starts generating antithetic variates, i.e., $1 - U$ instead of U, until the procedure is called again with `a = false`.

`public void increasedPrecis (boolean incp)`
After calling this procedure with `incp = true`, each call to the generator (direct or indirect) for this stream will return a uniform random number with more bits of resolution (53 bits

if machine follows IEEE-754 floating-point standard) instead of 32 bits, and will advance the state of the stream by 2 steps instead of 1. More precisely, if `s` is a stream of the class `RngStream`, in the non-antithetic case, the instruction "`u = s.randU01()`" will be equivalent to "`u = (s.randU01() + s.randU01(g) * fact) % 1.0`" where the constant `fact` is equal to 2^{-24}. This also applies when calling `randU01` indirectly (e.g., via `randInt`, etc.). By default, or if this procedure is called again with `incp = false`, each call to `randU01` for this stream advances the state by 1 step and returns a number with 32 bits of resolution.

`public void advanceState (int e, int c)`
Advances the state of this stream by k values, without modifying the states of other streams (as in `setSeed`), nor the values of B_g and I_g associated with this stream. If $e > 0$, then $k = 2^e + c$; if $e < 0$, then $k = -2^{-e} + c$; and if $e = 0$, then $k = c$. Note: c is allowed to take negative values. We discourage the use of this method.

`public boolean setSeed (long seed[])`
Sets the initial seed I_g of this stream to the vector `seed[0..5]`. This vector should contain valid seed values as described in `SetPackageSeed`. The state of the stream is then reset to this initial seed. The states and seeds of the other streams are not modified. As a result, after calling this method, the initial seeds of the streams are no longer spaced Z values apart. We discourage the use of this method; proper use of the `Reset` methods is preferable. Returns `false` for invalid seeds, and `true` otherwise.

`public double[] getState()`
Returns the current state C_g of this stream. This is a vector of 6 integers represented in floating point. This method is convenient if we want to save the state for subsequent use.

`public void writeState ()`
Prints (to standard output) the name and current state C_g of this stream.

`public void writeStateFull ()`
Prints (to standard output) the name of this stream and the values of all its internal variables.

`public double randU01 ()`
Returns a (pseudo)random number from the uniform distribution over the interval $(0, 1)$, using this stream, after advancing its state by one step. Normally, the returned number has 32 bits of resolution, in the sense that it is always a multiple of $1/(2^{32} - 208)$. However, if the precision has been increased by calling `increasedPrecis` for this stream, the resolution is higher and the stream state advances by two steps.

`public int randInt (int i, int j)`
Returns a (pseudo)random number from the discrete uniform distribution over the integers $\{i, i + 1, \ldots, j\}$, using this stream. Makes one call to `randU01`.

}

L'Ecuyer, P. 1999. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, **47**(1), 159–164.