The method of Algorithm 133 [1] satisfies Greenberger's condition, but since the reciprocal of its multiplying factor is as high as 0.2, Coveyou's result shows that it is very unsatisfactory for purposes requiring statistically independent consecutive random numbers.

Algorithms 133 and 266 have both been tested by computing a number of sets of 2000 successive random integers between 0 and 9, dividing each set into 400 groups of 5, and performing the poker test [4]. The results were classified in the following seven categories:

(*i*) all different
(*ii*) 1 pair
(*iii*) 2 pairs
(*iv*) 3 of a kind
(*v*) 3 of a kind and 1 pair
(*vi*) 4 of a kind
(*vii*) 5 of a kind.

The following tables resulted:

ALGORITHM 133

Run	*Starting Value*	(*i*)	(*ii*)	(*iii*)	(*iv*)	(*v*)	(*vi*)	(*vii*)
1	13421773	114	193	42	37	7	7	0
2	22369621	111	181	46	40	14	8	0
3	33554433	130	178	48	28	7	6	3
4	6871947673	118	179	51	35	10	5	2
5	11453246123	128	189	44	28	6	4	1
6	17179869185	135	155	45	52	6	5	2
Expected for each Run		120.96	201.60	43.20	28.80	3.60	1.80	0.04
Total for 6 Runs		736	1075	276	220	50	35	8
Expected for Total		725.76	1209.60	259.20	172.80	21.60	10.80	0.24

ALGORITHM 266

Run	*Starting Value*	(*i*)	(*ii*)	(*iii*)	(*iv*)	(*v*)	(*vi*)	(*vii*)
1	13421773	132	191	35	38	2	2	0
2	22369621	140	187	45	27	0	1	0
3	33554433	129	198	44	25	4	0	0
4	8426219	107	202	50	37	2	2	0
5	42758321	101	207	60	25	5	2	0
6	56237485	118	203	42	34	1	2	0
7	62104023	119	206	41	27	6	1	0
Expected for each Run		120.96	201.60	43.20	28.80	3.60	1.80	0.04
Total for 7 Runs		846	1394	317	213	20	10	0
Expected for Total		846.72	1411.20	302.40	201.60	25.20	12.60	0.28

Combining categories (*vi*) and (*vii*) in each case, the observed totals give χ^2 values (on 5 degrees of freedom) of 159.0 for Algorithm 133, and of 3.28 for Algorithm 266.

REFERENCES:

1. BEHRENZ, P. G. Algorithm 133, Random. *Comm. ACM 5* (NOV. 1962), 553.
2. COVEYOU, R. R. Serial correlation in the generation of pseudo-random numbers. *J. ACM 7*(1960), 72–74.
3. GREENBERGER, M. An a priori determination of serial correlation in computer generated random numbers. *Math. Comput. 15*(1961), 383–389. Correction in *Math. Comput.16*(1962), 126.
4. KENDALL, M. G., AND BABINGTON-SMITH, B. Randomness and random sampling numbers. *J. Royal Statist. Soc. 101* (1938), 147–166.

ALGORITHM 267
RANDOM NORMAL DEVIATE [G5]

M. C. PIKE (Recd. 3 May 1965 and 6 July 1965)
Medical Research Council, London, England


```
procedure RND(x1, x2, Random);
  real procedure Random; real x1, x2;
comment RND uses two calls of the real procedure Random
  which is any pseudo-random number generator which will
  produce at each call a random number lying strictly between 0
  and 1. A suitable procedure is given by Algorithm 266, Pseudo-
  Random Numbers [Comm. ACM 8(Oct. 1965), 605] if one chooses
  a = 0, b = 1 and initializes y to some large odd number, such as
  13421773. RND produces two independent random variables x1
  and x2 each from the normal distribution with mean 0 and
  variance 1. The method used is given by BOX, G.E.P., AND
  MULLER, M.E., A note on the generation of random normal
  deviates. [Ann. Math. Stat. 29 (1958), 610–611];
begin real t;
  x1 := sqrt(−2.0 × ln(Random));
  t := 6.2831853072 × Random;
  comment 6.2831853072 = 2 × pi;
  x2 := x1 × sin(t); x1 := x1 × cos(t)
end RND
```

Algorithm 121, NormDev [*Comm. ACM 5* (Sept. 1962), 482; *8* (Sept. 1965), 556] also produces random normal deviates and Algorithm 200, NORMAL RANDOM [*Comm. ACM 6* (Aug. 1963), 444; *8* (Sept. 1965), 556] produces random deviates with an approximate normal distribution, but the procedure *RND* seems preferable to both of them.

We may compare *NORMAL RANDOM* to *RND* (which is exact) by noting that at recommended minimum *n NORMAL RANDOM* requires 10 calls of *Random* while *RND* gets two independent normal deviates from 2 calls of *Random* and one call each of *sqrt*, *ln*, *sin* and *cos*. Under the stated test conditions a single call of *NORMAL RANDOM* (with $n = 10$) took 20 percent more computing time than a single call of *RND* when the real procedure *Random* was given by Algorithm 266.

To compare *NormDev* to *RND* in the same way, we have first to calculate the expected number of calls of *ln*, *sqrt*, *exp* and *Random* for each call of *NormDev*. This may be done by noting that there is (1) an initial single call of *Random*, then (2) with probability 0.68 a random normal deviate restricted to (0, 1) has to be found and this requires on average 1.36 calls of *Random* and 1.18 calls of *exp*, and (3) with probability 0.32 a random normal deviate restricted to $(1, \infty)$ has to be found and this requires on average 2.04 calls of *Random* and 1.52 calls of each of *ln* and *sqrt*. *NormDev* thus requires on average 2.58 calls of *Random*, 0.80 calls of *exp*, 0.49 calls of *ln* and 0.49 calls of *sqrt*. (Note: *NormDev* requires one further call of *Random* if a signed normal deviate is required.) Under the stated test conditions a single call of *NormDev* took virtually the same amount of computing time as a single call of *RND* when the real procedure *Random* was as above.

(Note: In testing *NormDev* the procedure was speeded up by replacing *A* by 0.6826894 wherever it occurred and removing it from the parameter list. In testing *NORMAL RANDOM Mean, Sigma, n* were replaced by 0, 1.0 and 10 respectively and removed from the parameter list.)