```
#include <stdlib.h>
#include <stdio.h>                                  LIBRARIES AND DEFINITIONS
#include <string.h>
#include <math.h>
#include <sys/types.h>
#include <time.h>

#define Nodes 20000
#define P 0.0025
/* if numbers of nodes N and edges E are given then p*N(N-1)/2 = E so p = 2E/N/(N-1) */
#define N 624
#define M 397
#define MATRIX_A 0x9908b0dfUL   /* constant vector a */
#define UPPER_MASK 0x80000000UL /* most significant w-r bits */
#define LOWER_MASK 0x7fffffffUL /* least significant r bits */
```

```
static unsigned long mt[N];     /* the array for the state vector  */
static int mti=N+1;             /* mti==N+1 means mt[N] is not initialized */

main(int argc, char *argv[]) {
  void init_genrand(unsigned long s); /* this is the seed of random number */
  unsigned long genrand_int32(void);
  double genrand_real2(void), p=0;
  int i, j, k, edges=0, nodes=Nodes;
  FILE *fout;                                   DECLARATIONS AND INITIALIZATION
  time_t rawtime;
  struct tm *info;

  time(&rawtime);
  info = gmtime(&rawtime);
  j = info->tm_min;
  k = info->tm_sec;
  for (i=1; i<j+k; i++) p = genrand_real2();
  p = P;

  if (argc>2)  sscanf(argv[2], "%d", &nodes);
  printf("nodes %d\n", nodes);
  if (argc>1)  sscanf(argv[1], "%lf", &p);
  printf("prob %lf\n", p);
  if (p<0 | p>1) {
    printf("Error: out of range probability, provide number between 0 and 1\n");
    exit(0);
  }

  /* open file */
  fout = fopen ("ERedges.txt", "w");
  if (fout==NULL) {
    printf("no output file\n");
    exit(-1);
  }
```

```
for (i=0; i<nodes; i++) for (j=i+1; j<nodes; j++) if (p>genrand_real2()) {
        edges++;
        fprintf(fout, "%d %d\n", i, j);                 MAIN LOOP
}
```

```
  fclose(fout);
  printf("produced %d edges\n", edges);                 CLOSING
```

```
}

/*---------------------------- Additional funtions -------------------------*/

void init_genrand(unsigned long s) {                    RANDOM NUMBER GENERATOR

  mt[0]= s & 0xffffffffUL;
  for (mti=1; mti<N; mti++) {
    mt[mti] = (1812433253UL * (mt[mti-1] ^ (mt[mti-1] >> 30)) + mti);
    mt[mti] &= 0xffffffffUL;
  }
}

unsigned long genrand_int32(void) {
  unsigned long y;
  static unsigned long mag01[2]={0x0UL, MATRIX_A};
  if (mti >= N) {
    int kk;

    if (mti == N+1) init_genrand(5489UL);

    for (kk=0;kk<N-M;kk++) {
      y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
      mt[kk] = mt[kk+M] ^ (y >> 1) ^ mag01[y & 0x1UL];
    }
    for (;kk<N-1;kk++) {
      y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
      mt[kk] = mt[kk+(M-N)] ^ (y >> 1) ^ mag01[y & 0x1UL];
    }
    y = (mt[N-1]&UPPER_MASK)|(mt[0]&LOWER_MASK);
    mt[N-1] = mt[M-1] ^ (y >> 1) ^ mag01[y & 0x1UL];

    mti = 0;
  }

  y = mt[mti++];
  y ^= (y >> 11);
  y ^= (y << 7) & 0x9d2c5680UL;
  y ^= (y << 15) & 0xefc60000UL;
  y ^= (y >> 18);

  return y;
}

double genrand_real2(void) {
    return genrand_int32()*(1.0/4294967296.0);
}
```