```
/*********************************************************************/
/*** Generator for Binomial distributed numbers                    ***/
/*** Lectures in computational physics                             ***/
/*** Universities of Goettingen + Oldenburg, Germany 2004-    ***/
/*** compile with: cc -o bernoulli bernoulli_direct.c -lm     ***/
/*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int main(int argc, char *argv[])
{
  int n;                                  /* number of values */
  int num_bins;                           /* number of bins */
  double *histo;                          /* histogram */
  double start_histo, end_histo;          /* range of histogram */
  double delta;                           /* width of bin */
  int bin;                                /* ID of current bin */
  int t ,t2;                              /* loop counters */
  int num_runs;                 /* number of generated random numbers */
  double alpha;                           /* parameter of distribution */
  double value;                           /* generated number */

  if(argc != 4)
  {
    fprintf(stderr, "USAGE: %s <n> <alpha> <num_runs>\n", argv[0]);
    exit(1);
  }

  n = atoi(argv[1]);                      /* read parameters */
  sscanf(argv[2], "%lf", &alpha);
  num_runs = atoi(argv[3]);

  num_bins = n+1;
  histo = (double *) malloc( num_bins*sizeof(double));

  for(t=0; t< num_bins; t++)              /* initialise histogram */
     histo[t] = 0;
  start_histo = -0.5; end_histo = n+0.5;
  delta = (end_histo - start_histo)/num_bins;

  for(t=0; t<num_runs; t++)                       /* main loop */
  {
    value = 0;
    for(t2=0; t2<n; t2++)
      if(drand48() < alpha)
        value++;

    bin = (int) floor((value-start_histo)/delta);
    if( (bin >= 0)&&(bin < num_bins))           /* inside range ? */
      histo[bin]++;                           /* count event */
  }

  for(t=0; t<num_bins; t++)             /* print normalized histogram */
     printf("%f %e\n", start_histo + (t+0.5)*delta,
           histo[t]/(delta*num_runs));

  free(histo);

  return(0);
}
```