```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "nr.h"
#include "nrutil.h"

int main()
{
    FILE *filename, *filename2, *filename3, *filename4, *filename5;
    char name_file[150], name_file2[150], name_file3[150], name_file4[150],
        name_file5[150];

    long idum = -10;
    int s, p, t, temp, temp_a, ini, x_loc;
    double dt, frac;
    double bc_limit=1.0, bin=0.05, tot=0.0, n_x_temp;
    //double dt=pow(bin,2)/20.;
    int N_s = ( bc_limit-(-1.0*bc_limit) )/bin;
    int N_t=10000, N_p=11000, N_pa=N_p, ratio=200;

    //double **x = malloc (N_t * sizeof (double *));
    //double **x_a = malloc (N_t * sizeof (double *));
    //int **n = malloc (N_t * sizeof (int *));
    //int **n_a = malloc (N_t * sizeof (int *));
    //int **n_tot = malloc (N_t * sizeof (int *));
    //int **D = malloc (N_t * sizeof (int *));
    //double *x_coord = malloc (N_s * sizeof (double));
    //double *t_coord = malloc (N_t * sizeof (double));
    //double *q = malloc (N_p * sizeof (double));

    double x[2][N_p+1];
    double x_a[2][N_pa+1];
    int n[N_t+1][N_s+1];
    int n_a[N_t+1][N_s+1];
    int n_tot[N_t+1][N_s+1];
    double x_coord[N_s+1];
    double t_coord[N_t+1];
    double u0[N_s+1];
    int D[N_s+1];
    int D_a[N_s+1];
    double q, delta_x;


    /*
    for (t=0;t<=N_t;t++){
        x[t] = malloc (N_p * sizeof (double));
        x_a[t] = malloc (N_p * sizeof (double));
        //n[t] = malloc (N_s * sizeof (int));
        //n_a[t] = malloc (N_s * sizeof (int));
        //n_tot[t] = malloc (N_s * sizeof (int));
        //D[t] = malloc (N_s * sizeof (int));
    }
    */

    // x-coord & u0
    for (s=0;s<=N_s;s++){
        x_coord[s] = bin * s + (-1.0 * bc_limit) ;
        u0[s] = x_coord[s] +1.0 ;
```

```
            tot += u0[s];
        }

        // t-coord
        t_coord[0]=0.0;
        //for (t=0;t<=N_t;t++) t_coord[t] = dt * t;

        // initialization
        for (t=0;t<=N_t;t++){
            for (s=0;s<=N_s;s++){
                n[t][s] = 0;
                n_a[t][s] = 0;
                n_tot[t][s] = 0;
            }
            /*
            for (p=0;p<=N_p;p++){
                x[0][p] = 0.0;
                x[1][p] = 0.0;
                x_a[0][p] = 0.0;
                x_a[1][p] = 0.0;
            }
            */
        }

        // initial condition
        for (s=0;s<=N_s;s++){
            n[0][s] = ratio;
            frac = (x_coord[s]+bc_limit)/(x_coord[N_s]+bc_limit);
            n_a[0][s] = ratio*2 *( 1-frac ) ;
            n_tot[0][s] = n[0][s] - n_a[0][s];
            printf("%f\t%d\t%d\t%d\n",frac,n[0][s],n_a[0][s],n_tot[0][s]);
        }


        int tt=0;
        //Monte Carlo
        for (t=1;t<=N_t;t++){
            int k_ini=0, k=0, ka_ini=0, ka=0;

            double dt_temp, min=99999.0;
            for (s=1;s<=N_s-1;s++){
                dt_temp = (n[t-1][s+1]-n[t-1][s-1])/2.0/bin;
                dt_temp = 1./( 48. * dt_temp * dt_temp );
                if (dt_temp < min) min=dt_temp;
                dt_temp = (n_a[t-1][s+1]-n_a[t-1][s-1])/2.0/bin;
                dt_temp = 1./( 48. * dt_temp * dt_temp );
                if (dt_temp < min) min=dt_temp;
            }
            dt = 0.5 * min;
            //dt = 2.0 * min;
            //dt = 0.00001;
            t_coord[t] = t_coord[t-1]+dt;


            // for + paticle
            for (s=0;s<=N_s;s++){

                // n to x
                temp =0;
                k_ini = k;
                while (temp < n[t-1][s]){
                    x[t-tt-1][k] = x_coord[s];
                    //printf("%d\t%d\t%d\t%f\t%f\n",s, n[0]
                        [s],temp,x_coord[s],x[0][k]);
                    temp++;
                    k++;
                    //printf("%d\n",k);
                }
                //printf("k_ini: \t");
                //printf("%d\t%d\t%d\n",s ,temp ,k_ini);


                // evolve x
                D[s] = 3* pow(n_tot[t-1][s],2);

                for (p=k_ini;p<k;p++){
                    q = gasdev(&idum);
                    delta_x = q * sqrt( 2*D[s]*dt );
                    x_loc = round( ( x[t-tt-1][p]+delta_x + bc_limit ) / bin );

                    if (x_loc >= 0 && x_loc<= N_s){
                        x[t-tt][p] = x_coord[ x_loc ];

                        // x to n
                        n[t][x_loc]++;

                        //printf("particle:\t");
                        //printf("%d\t%d\t%f\t%d\t%f\t%f\t%d\n", t, p, delta_x,
                            x_loc, x[t-tt-1][p], x[t-tt][p], n[t][x_loc]);
                        //printf("%d\n",N_p);

                    }

                }
                //printf("chk");


            }

            // for anti - paticle
            for (s=0;s<=N_s;s++){

                // n to x
                temp_a =0;
                ka_ini = ka;
                while (temp_a < n_a[t-1][s]){
                    x_a[t-tt-1][ka] = x_coord[s];
                    //printf("%d\t%d\t%d\t%f\t%f\n",s, n[0]
                        [s],temp,x_coord[s],x[0][k]);
                    temp_a++;
                    ka++;
                    //printf("%d\n",k);
                }
                //printf("k_ini: \t");
                //printf("%d\t%d\t%d\n",s ,temp ,k_ini);


                // evolve x
                D_a[s] = 3* pow(n_tot[t-1][s],2);

                for (p=ka_ini;p<ka;p++){
                    q = gasdev(&idum);
                    delta_x = q * sqrt( 2*D_a[s]*dt );
                    x_loc = round( ( x_a[t-tt-1][p]+delta_x + bc_limit ) / bin );

                    if (x_loc >= 0 && x_loc<= N_s){
                        x_a[t-tt][p] = x_coord[ x_loc ];

                        // x to n
                        n_a[t][x_loc]++;

                        //printf("anti-particle:\t");
                        //printf("%d\t%d\t%f\t%d\t%f\t%f\t%d\n", t, p, delta_x,
                            x_loc, x_a[t-tt-1][p], x_a[t-tt][p], n_a[t][x_loc]);
                        //printf("%d\n",N_pa);


                    }

                    /*
                    else {
                        x_loc = s;
                        x[t-tt][p] = x_coord[ s ];

                        n[t][x_loc]++;

                    }
                    */

                }



            }
            tt=t;


            // bc
            n[t][0] = n[0][0];
            n[t][N_s] = n[0][N_s];
            n_a[t][0] = n_a[0][0];
            n_a[t][N_s] = n_a[0][N_s];

            // number of particle
            N_p=0;
            N_pa=0;
            for (s=0;s<=N_s;s++){
                N_p+=n[t][s];
                N_pa+=n_a[t][s];
            }

            double x[2][N_p+1];
            double x_a[2][N_pa+1];

            for (s=0;s<=N_p;s++){
                x[0][s]=-9999.;
                x[1][s]=-9999.;
            }
            for (s=0;s<=N_pa;s++){
                x_a[0][s]=-9999.;
                x_a[1][s]=-9999.;
            }


            //for (s=0;s<=N_s;s++) printf("%d\t%d\t%d\t%f\n", 0, s, n_tot[0][s],
                x_coord[s]);

            for (s=0;s<=N_s;s++){
                n_tot[t][s] = n[t][s] - n_a[t][s];
                //printf("%d\t%d\t%d\t%d\t%d\t%f\n", t, s, n[t][s], n_a[t][s],
                    n_tot[t][s], x_coord[s]);
            }


        }


        // output data
        strcpy (name_file, "/Users/natii/nati/UIUC/2014Fall/CPA/ps9/q3c/data/
            n.dat");
        filename = fopen (name_file, "w");

        for (s=0;s<=N_s;s++) {
            for (t=0;t<=N_t;t++){
                fprintf(filename, "%d\t", n_tot[t][s]);
            }
            fprintf(filename, "\n");
        }
        fclose (filename);


        strcpy (name_file2, "/Users/natii/nati/UIUC/2014Fall/CPA/ps9/q3c/data/
            t.dat");
        filename2 = fopen (name_file2, "w");

        for (t=0;t<=N_t;t++) {
            fprintf(filename2, "%d\t%f\n",t ,t_coord[t]);
            //printf("%f\t",t[i]);
        }
        fclose (filename2);

        strcpy (name_file3, "/Users/natii/nati/UIUC/2014Fall/CPA/ps9/q3c/data/
            x.dat");
        filename3 = fopen (name_file3, "w");

        for (s=0;s<=N_s;s++) {
            fprintf(filename3, "%d\t%f\n",s ,x_coord[s]);
        }
        fclose (filename3);

        strcpy (name_file4, "/Users/natii/nati/UIUC/2014Fall/CPA/ps9/q3c/data/
            n_pos.dat");
        filename4 = fopen (name_file4, "w");

        for (s=0;s<=N_s;s++) {
            for (t=0;t<=N_t;t++){
                fprintf(filename4, "%d\t", n[t][s]);
            }
            fprintf(filename4, "\n");
        }
        fclose (filename4);

        strcpy (name_file5, "/Users/natii/nati/UIUC/2014Fall/CPA/ps9/q3c/data/
            n_neg.dat");
        filename5 = fopen (name_file5, "w");

        for (s=0;s<=N_s;s++) {
            for (t=0;t<=N_t;t++){
                fprintf(filename5, "%d\t", n_tot[t][s]);
            }
            fprintf(filename5, "\n");
        }
        fclose (filename5);




        //free (x);
        //free (x_a);
        //free (n);
        //free (n_a);
        //free (n_tot);
        //free (D);
        //free (x_coord);
        //free (t_coord);
        //free (q);



        return 0;

    }

    /*
    int XtoN (int N_p, int N_s, double **x ){


    }
    */

    /*
    int NtoX ( int t, int N_p, int N_s, int **n, double x_coord[], double
        *x_temp ){

        double *xx = malloc (N_p * sizeof (double));
        int s, p, temp, k=0;
```

```
        for(p=0;p<=N_p;p++) xx[p]=1.0;
        for(s=0;s<=N_s;s++){
            temp =0;
            while (temp < n[t][s]){
                xx[k] = x_coord[s];
                x_temp[k] = x_coord[s];
                //printf("%d\t%d\t%d\t%f\t%f\n",s, n[t][s],temp,x_coord[s],xx[k]);
                temp++;
                k++;

                //printf("%f\n",xx[k]);
            }
        }
        // *x_temp = *xx;
        //free (xx);

        return 0;
    }
    */
```