void sim1(int init[], int initot, double rm, double mu, int \*freq\_arr[], int \*val\_arr[], int \*noall)

{

int j, nmax, ic, k;

float rr;

NEXTMUT = 0;

\*noall = 0;

for (j = 0; j<Spno; ++j){

Sd[j] = Spno / 0.5;

Migrate[j] = Sd[j] \* rm;

}

Occ = Subs;

for (j = 0; j<Spno; ++j)Occlist[j] = 0;

for (j = 0; j<Subs; ++j){

Occlist[j] = j;

}

for (j = 0; j<Spno; ++j){

Ni[j] = init[j];

}

Ntot = initot;

nmax = 10 \* Ntot;

Nlist = (struct node \*\*)malloc(nmax\*sizeof(struct node \*));

ic = 0;

for (k = 0; k<Spno; ++k){

Lmax[k] = 2 \* init[k];

if (Lmax[k] < 10)Lmax[k] = 10;

}

for (k = 0; k<Spno; ++k){

List[k] = (struct node \*\*)malloc(Lmax[k] \* sizeof(struct node \*));

for (j = 0; j<Ni[k]; ++j){

List[k][j] = (struct node \*)malloc(sizeof(struct node));

List[k][j]->d[0] = List[k][j]->d[1] = NULL;

List[k][j]->a[0] = List[k][j]->a[1] = NULL;

List[k][j]->time = 0.0;

List[k][j]->dna = 0;

List[k][j]->I = 0;

List[k][j]->sp = Occlist[k];

List[k][j]->osp = Occlist[k];

Nlist[ic] = List[k][j];

++ic;

}

}

N\_n = Ntot;

Tt = 0.0;

while (1){

if (Occ > Spno){

printf("error Occ > Spno\n");

exit(1);

}

for (k = 0; k<Occ; ++k){

if (Ni[k] > Lmax[k]){

printf("error in Ni/Lmax\n");

exit(1);

}

if (Ni[k] >= Lmax[k] - 5){

Lmax[k] = 2 \* Lmax[k];

if (Ni[k] > Lmax[k]){

printf("error - Lmax");

exit(1);

}

for (j = 0; j<Ni[k]; ++j){

if (List[k][j]->sp != Occlist[k]){

printf("error in sp \n");

exit(1);

}

}

List[k] = (struct node \*\*)realloc(

List[k], Lmax[k] \* sizeof(struct node \*));

for (j = 0; j<Ni[k]; ++j){

if (List[k][j]->sp != Occlist[k]){

printf("error in sp \n");

exit(1);

}

}

}

}

if (N\_n >= nmax - 1){

nmax = 2 \* nmax;

Nlist = (struct node \*\*)realloc(

Nlist, nmax\*sizeof(struct node \*));

}

dfill();

Tt += expdev() / Dtop;

loopback: rr = gfsr4();

for (k = 0; k<Occ; ++k){

for (j = 0; j<2; ++j){

if (rr < Den[k][j] / Dtop)goto loopout;

}

}

goto loopback;

loopout: if (j == 0){

cnode(k);

if (Ntot == 1)break;

}

else mnode(k);

}

Nlist[N\_n - 1]->dna = 0;

\*noall = 0;

for (j = N\_n - 1; j >= 0; --j){

treefill(Nlist[j], noall, freq\_arr, val\_arr, mu);

}

for (j = 0; j<N\_n; ++j)killtree(Nlist[j]);

for (j = 0; j<Spno; ++j)free(List[j]);

free(Nlist);

}