```
//////////////////////
//
//  www.maknoon.com
//
//////////////////////

#include <FL/fl_draw.H>
#include <FL/fl_ask.H>
#include <FL/Fl.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Round_Button.H>
#include <FL/Fl_Int_Input.H>
#include <FL/Fl_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Widget.H>
#include <FL/Fl_Group.H>
#include <FL/Fl_Float_Input.H>
#include <FL/Fl_Value_Output.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Gl_Window.H>
#include <FL/Fl_BMP_Image.H>
#include <FL/Fl_Menu_Button.H>
#include <FL/Fl_Image.H>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <iomanip>


using namespace std;

void intializesystem();
int drawinterface();
void getuserinputRadio(Fl_Widget *w, void *v);
void getuserinputArrival(Fl_Widget *w, void *v);
void getuserinputAvgDep(Fl_Widget *w, void *v);
void getuserinputRunTime(Fl_Widget *w, void *v);
int poisson(double lambda);
void increaseCar(int iteration);
void simulateQ(Fl_Widget *w, void *v);
void decreaseCar(int );
void trafficLight(int );
void showOut();
double sample_normal(double mean, double std_dev);
void MAXMINq (int ,int);
void MAXMINwt (int ,int);
void demoCB (Fl_Widget *w, void *v);
int checkInput();
void reinitCB(Fl_Widget *w,void *v);
void calstat(int loops);
void tablegen();
void demobut(Fl_Widget *w, void *v);


int arrivalrate=-1,runtime,NofLanes,increaseStat=0,openLane[8],MaxLane=0,gSEQ[8]={0},LopenLane[
8]={0}
,q[]={0,2,4,6},GreenLDuration=0,lastY=66,SDDT=1,draw=0,drawn[8],numbofCars[8]={0}
,divloop,MAXq[8]={0},MINq[8]={35345};
double AvgDepT=-1,timeInterval,Qwt[8]={1},MAXwt[8]={0},MINwt[8]={2222},Gdeptime[8],Wavg[8]={0}
,Qavg[8]={0},Qlengths[8]={0},decreseStats=0;
const char *temp;


struct carT {
    int colorR ;
    int colorG ;
    int colorB ;
    int timeR;
    double deptime;
    carT* next;
}car,*currentCarQ1,*firstCarQ1,*q1,*currentCarQ2,*firstCarQ2,*q2,*currentCarQ3,*firstCarQ3
,*q3,*currentCarQ4,*firstCarQ4,*q4,*currentCarQ5,*firstCarQ5,*q5,*currentCarQ6,*firstCarQ6
,*q6,*currentCarQ7,*firstCarQ7,*q7,*currentCarQ8,*firstCarQ8,*q8;
```

```
int Nq1=0,Nq2=0,Nq3=0,Nq4=0,Nq5=0,Nq6=0,Nq7=0,Nq8=0;
char TLS[8][1000]={0};

Fl_Value_Output *outQ[8],*time1;
Fl_Box *Tlight[8];
Fl_Double_Window *window;
Fl_Box *test[9][15];
Fl_Group *boxs,*Group2,*Group,*Grun;
Fl_Group *GroupLanes,*GroupTInterval;
Fl_Button *run;
Fl_Round_Button *onesecinterval;
Fl_Round_Button *tensecinterval;
Fl_Round_Button *onemininterval;
Fl_Round_Button *onelane;
Fl_Round_Button *twolane;
Fl_Int_Input *arrivalinput;
Fl_Float_Input *runningtimeinput;
Fl_Float_Input *avgdepinput;
Fl_Box *TimeIntervaltext;
Fl_Box *NumberOfLanestext;
Fl_Button *Exitt=(Fl_Button *)0;
Fl_Button *Reinit=(Fl_Button *)0;
Fl_Button *Run=(Fl_Button *)0;
Fl_Menu_Button *Demosub=(Fl_Menu_Button *)0;
Fl_Menu_Item menu_Demosub[] = {
 {"Demo 1", 0,  (Fl_Callback *)demobut, (void*)1, 0, 0, 0, 14, 56},
 {"Demo 2", 0,  (Fl_Callback *)demobut, (void*)2, 0, 0, 0, 14, 56},
 {"Demo 3", 0,  (Fl_Callback *)demobut, (void*)3, 0, 0, 0, 14, 56},
 {0}
};


//////////////////////

inline ifstream& eatwhite2(ifstream& is)
{
char c;
while(is.get(c)) {

    if (!isspace(c)) {
      is.putback(c);
      break;
    }

}
    return is;

}

//////////////////////

void exitrt(Fl_Widget *w, void *v){
    exit(0);
}

//////////////////////
int main(){

    intializesystem();

    drawinterface();

    return Fl::run();


}

////////////////////////////////



void intializesystem(){
```

```
ifstream in ("TLS.txt",ios::in);
ofstream tests("test.txt",ios::out);



int y=0;

while(!(in.eof())){

    eatwhite2(in);




    for(int x=0;x<8;x++){

        in.get(TLS[x][y]);
        //tests << TLS[x][y];


    }

    y++;
    //tests<<endl;
    eatwhite2(in);
}

MaxLane=y;



tests.close();
in.close();



    q1 = new (struct carT);

    *q1 = car;
    currentCarQ1=q1;
    firstCarQ1 = q1;
    firstCarQ1->colorB=rand()%256;


    q2 = new (struct carT);

    *q2 = car;
    currentCarQ2=q2;
    firstCarQ2 = q2;
    firstCarQ2->colorB=rand()%256;


    q3 = new (struct carT);

    *q3 = car;
    currentCarQ3=q3;
    firstCarQ3 = q3;
    firstCarQ3->colorB=rand()%256;


    q4 = new (struct carT);

    *q4 = car;
    currentCarQ4=q4;
    firstCarQ4 = q4;
    firstCarQ4->colorB=rand()%256;


    q5 = new (struct carT);

    *q5 = car;
    currentCarQ5=q5;
    firstCarQ5 = q5;
    firstCarQ5->colorB=rand()%256;


    q6 = new (struct carT);
```

```
        *q6 = car;
        currentCarQ6=q6;
        firstCarQ6 = q6;
        firstCarQ6->colorB=rand()%256;


        q7 = new (struct carT);

        *q7 = car;
        currentCarQ7=q7;
        firstCarQ7 = q7;
        firstCarQ7->colorB=rand()%256;


        q8 = new (struct carT);

        *q8 = car;
        currentCarQ8=q8;
        firstCarQ8 = q8;
        firstCarQ8->colorB=rand()%256;


}



//////////////////////////////////////////////////

int drawinterface(){


        char *Road_Label[4]={"Road 1:","Road 2:","Road 3:","Road 4:"};

        window= new Fl_Double_Window(700,550,"SimuTraffic");
        Fl_BMP_Image *imageT=new Fl_BMP_Image("sim.bmp");
        Fl_Box *image_box=new Fl_Box(0,0,502,60);
        Fl_Box *road1=new Fl_Box(145,75,10,20,"Road 1 ->>");
        Fl_Box *road2=new Fl_Box(310,245,10,20,"Road 2");
        Fl_Box *road3=new Fl_Box(300,375,10,20,"<<- Road 3");
        Fl_Box *road4=new Fl_Box(135,205,10,20,"Road 4");
        time1 =new Fl_Value_Output(580,20,55,20,"Time(secs):");
        Fl_Box *DrawBox= new Fl_Box(60,70,330,330);
        Fl_Box *LeftTopbox= new Fl_Box(60,70,130,130);
        Fl_Box *RightTopbox= new Fl_Box(260,70,130,130);
        Fl_Box *LeftBotombox= new Fl_Box(60,270,130,130);
        Fl_Box *RightBotombox= new Fl_Box(260,270,130,130);

        for(int i=0;i<4;i++)
                  {
              outQ[i]=new Fl_Value_Output(60+(i%2==0?(i*100):((i-1)*145) +(i==1?90:0)),440,55,2
0);
                      outQ[i]->hide();
                      if(i%2 == 0) {
                          outQ[i]->label(Road_Label[(i==0)?i:i-1]);
                          outQ[i]->align(FL_ALIGN_TOP && FL_ALIGN_LEFT);
                      }
                      outQ[i]->value(i);

                  }


          for(i=4;i<8;i++)
                  {
              outQ[i]=new Fl_Value_Output(60+((i-4)%2==0?((i-4)*100):(((i-4)-1)*145) +((i-4)==1
?90:0)),490,55,20);
                      outQ[i]->hide();
                      if(i%2 == 0) {
                          outQ[i]->label(Road_Label[(i==4)?(i-2):(i-3)]);
                          outQ[i]->align(FL_ALIGN_TOP && FL_ALIGN_LEFT);
                      }
                      outQ[i]->value(i);
                  }
```

```
  { Fl_Button* o = Exitt = new Fl_Button(455, 465, 80, 25, "Exit");
 o->box(FL_PLASTIC_UP_BOX);
 o->labeltype(FL_ENGRAVED_LABEL);

 o->callback(exitrt);
 o->color(1);
   }

   Group2= new Fl_Group(400,0,700,500);

    { Fl_Button* o = Reinit = new Fl_Button(455, 435, 80, 25, "Reintialize");
 o->box(FL_PLASTIC_UP_BOX);
o->color(92);
 o->labeltype(FL_ENGRAVED_LABEL);

    }


   Grun=new Fl_Group(400,0,700,500);
    { Fl_Button* o = Run = new Fl_Button(455, 405, 80, 25, "Run");
 o->box(FL_PLASTIC_UP_BOX);
 o->labeltype(FL_ENGRAVED_LABEL);
 o->color(110);

    }

    { Fl_Menu_Button* o = Demosub = new Fl_Menu_Button(540, 405, 80, 25, "Demo");
 o->box(FL_PLASTIC_UP_BOX);
 o->labeltype(FL_ENGRAVED_LABEL);
 o->align(FL_ALIGN_LEFT|FL_ALIGN_INSIDE);
 o->menu(menu_Demosub);
    }

    Grun->end();



  onesecinterval= new Fl_Round_Button(450,100,100,20,"0.001 Sec");
  tensecinterval= new Fl_Round_Button(450,125,100,20,"0.01 Sec");
  onemininterval= new Fl_Round_Button(570,100,70,20,"0.1 Sec");
  onelane= new Fl_Round_Button(450,270,90,20,"One Lane");
  twolane= new Fl_Round_Button(450,295,90,20,"Two Lanes");
  arrivalinput= new Fl_Int_Input(550,155,70,20,"Arrival Rate/min      ");
  runningtimeinput= new Fl_Float_Input(550,199,70,20,"Running time (mins)");
  avgdepinput= new Fl_Float_Input(600,330,70,20,"Avg. Dep. Time(secs)/car   ");
  TimeIntervaltext= new Fl_Box(415,35,100,100,"Time Interval :");
  NumberOfLanestext= new Fl_Box(430,230,100,30,"Number Of Lanes :");
  GroupTInterval= new Fl_Group(445,95,220,55);
  GroupLanes= new Fl_Group(445,268,110,50);




  //Fl_Box *road1_label=new Fl_Box(60,400,10,20,"Road 1:");
  //Fl_Box *road2_label=new Fl_Box(310,245,10,20,"Road 2");
  //Fl_Box *road3_label=new Fl_Box(300,375,10,20,"<<- Road 3");
  //Fl_Box *road4_label=new Fl_Box(135,205,10,20,"Road 4");









          /*Fl_Round_Button *test= new Fl_Round_Button(20,20,90,20,"test");
          test->selection_color(11);
          Group->add(test);*/
```

```
road1->labelcolor(255);
road1->labelfont(FL_HELVETICA_BOLD);
road2->labelcolor(255);
road2->labelfont(FL_HELVETICA_BOLD);
road3->labelcolor(255);
road3->labelfont(FL_HELVETICA_BOLD);
road4->labelcolor(255);
road4->labelfont(FL_HELVETICA_BOLD);



srand( (unsigned)time(NULL));
DrawBox->color(FL_DARK2);
DrawBox->box(FL_FLAT_BOX);
LeftTopbox->color(((rand()%10)>5)?60:58);
LeftTopbox->box(FL_UP_BOX);
RightTopbox->color(((rand()%10)>5)?60:58);
RightTopbox->box(FL_UP_BOX);
LeftBotombox->color(((rand()%10)>5)?60:58);
LeftBotombox->box(FL_UP_BOX);
RightBotombox->color(((rand()%10)>5)?60:58);
RightBotombox->box(FL_UP_BOX);

GroupTInterval->add(onesecinterval);
GroupTInterval->add(tensecinterval);
GroupTInterval->add(onemininterval);
GroupTInterval->box(FL_DOWN_BOX);

onesecinterval->type(FL_RADIO_BUTTON);
tensecinterval->type(FL_RADIO_BUTTON);
onemininterval->type(FL_RADIO_BUTTON);


GroupLanes->box(FL_DOWN_BOX);
GroupLanes->add(onelane);
GroupLanes->add(twolane);

onelane->type(FL_RADIO_BUTTON);
twolane->type(FL_RADIO_BUTTON);



image_box->image(imageT);

Group2->add(GroupTInterval);
Group2->add(GroupLanes);


//////////

onesecinterval->when(FL_WHEN_CHANGED);
tensecinterval->when(FL_WHEN_CHANGED);
onemininterval->when(FL_WHEN_CHANGED);
onelane->when(FL_WHEN_CHANGED);
twolane->when(FL_WHEN_CHANGED);
arrivalinput->when(FL_WHEN_CHANGED);
runningtimeinput->when(FL_WHEN_CHANGED);
avgdepinput->when(FL_WHEN_CHANGED);



onesecinterval->callback(getuserinputRadio,(void*) 1);
tensecinterval->callback(getuserinputRadio,(void*) 2);
onemininterval->callback(getuserinputRadio,(void*) 3);

onelane->callback(getuserinputRadio,(void*) 4);
twolane->callback(getuserinputRadio,(void*) 5);

arrivalinput->callback(getuserinputArrival,(void*) arrivalinput->value());
runningtimeinput->callback(getuserinputRunTime,(void*) 7);
avgdepinput->callback(getuserinputAvgDep,(void*) 8);


Run->callback(simulateQ);
Reinit->callback(reinitCB);
```

```
        /////Car Boxes////

        //Fl_Box *Q1car1= new Fl_Box(220,228,25,15);
    //  Q1car1->color(FL_BLUE);
    //  Q1car1->box(FL_UP_BOX);

        //Fl_Box *RightBotombox= new Fl_Box(220,250,130,130);
        //Fl_Box *RightBotombox= new Fl_Box(220,250,130,130);




        //onesecinterval->setonly();
        //tensecinterval->setonly();
        //oneminterval->setonly();









        Group2->end();
        Group = new Fl_Group(0,0,400,500);


                for(i=0;i<2;i++)
                  {
                         Tlight[i]=new Fl_Box((191+(21*i)),170,10,10);
                         Tlight[i]->hide();
                         Tlight[i]->box(FL_ROUNDED_BOX);
                  }

                for(i=2;i<4;i++)
                  {
                         Tlight[i]=new Fl_Box(270,(160+(20*i)),10,10);
                         Tlight[i]->hide();
                         Tlight[i]->box(FL_ROUNDED_BOX);
                  }

                for(i=4;i<6;i++)
                  {
                         Tlight[i]=new Fl_Box((250-(21*(i-4))),280,10,10);
                         Tlight[i]->hide();
                         Tlight[i]->box(FL_ROUNDED_BOX);
                  }

                for(i=6;i<8;i++)
                  {
                         Tlight[i]=new Fl_Box(160,(379-(20*i)),10,10);
                         Tlight[i]->hide();
                         Tlight[i]->box(FL_ROUNDED_BOX);



                  }



        window->end();
        window->show();


        return 0;


}


////////////////////////////////////////////////////
void getuserinputRadio(Fl_Widget *w, void *v){


    ofstream tests("test.txt",ios::app);
```

```
      if( (int)v == 1)
      {
          timeInterval=0.001;
      tests <<timeInterval<<endl;
      }

      if( (int)v == 2)
      {
          timeInterval=0.01;
      tests <<timeInterval <<endl;
      }

      if( (int)v == 3)
      {
          timeInterval=0.1;
      tests << timeInterval<<endl;
      }

      if( (int)v == 4)
      {   NofLanes=1;



          if(draw == 0){
              for(int i=0;i<4;i++)
              {
                  outQ[q[i]]->show();
                  Tlight[q[i]]->show();
                  window->redraw();

              }
              draw=1;
          }

          else if(draw == 2)
          {
              for(int i=0;i<7;i+=2)
              {
                  outQ[i+1]->hide();
                  Tlight[i+1]->hide();
                  window->redraw();

              }draw =1;

          }



      }

      if( (int)v == 5)
      {  NofLanes=2;
          if(draw == 1 || draw == 0){
                  for(int i=0;i<8;i++)
                  {
                      outQ[i]->show();
                      Tlight[i]->show();
                      window->redraw();
                  }
                  draw=2;
              }


      }


 tests.close();

}

//////////////////////////////////////////

void getuserinputArrival(Fl_Widget *w, void *v){

   ofstream tests("test.txt",ios::app);
```

```
   arrivalrate=atoi(((Fl_Int_Input*)w)->value());

   tests <<arrivalrate<<endl;

   tests.close();

}

//////////////////////////////////////////

void getuserinputAvgDep(Fl_Widget *w, void *v){


   ofstream tests("test.txt",ios::app);

   temp=((Fl_Float_Input*)w)->value();

   AvgDepT=atof(temp);

   tests <<AvgDepT<<endl;

   tests.close();


}

//////////////////////////////////////////
void getuserinputRunTime(Fl_Widget *w, void *v){

     ofstream tests("test.txt",ios::app);

   runtime=(atof(((Fl_Float_Input*)w)->value()))*60;

   tests <<runtime<<endl;

   tests.close();

}

//////////////////////////////////////////////

int poisson(double lambda){

        int count;
        static int total=0;
        double product;
        double zero_probability;
        ofstream tests;
        int debug=0;

        if (debug)
            tests.open("test.txt",ios::app);

        // lambda is the average arrival rate per minute
        // however, the unit time for the simulation is seconds
        // therefore, we need to convert lambda to arrival rate per second
        // we then multiply by the time interval to produce the arrival rate per time interval


        lambda=timeInterval*lambda/60;


        count = 0;
        product = (double) rand () / (double) RAND_MAX;

        zero_probability = exp(-lambda);

        while (product > zero_probability)
        {
            count++;
            product = product * ((double) rand() / (double) RAND_MAX);
        }


        if (debug)
            tests << count <<" count" <<endl;
```

```
        if (debug)
            tests.close();

        //intf("%d\r",total+=count);

        return (count);

}

//////////////////////////////////////////

int checkInput(){

    //printf("%d\n",arrivalinput->value());
    //printf("%d\n",avgdepinput->value());
    //printf("%d\n",runningtimeinput->value());

    if(arrivalrate == -1 || NofLanes == 0 || runtime == 0 || timeInterval == 0 || AvgDepT ==-1
)
    return 1;

    else return 0;

}

//////////////////////////////////////////


void simulateQ(Fl_Widget *w, void *v){

ofstream tests;
tests.open("test.txt",ios::app);
//tests.setf(ios::showpoint);
//tests.precision(19);
    srand( (unsigned)time(NULL));
    double Time=time(NULL);

    if(checkInput() != 0)
        fl_alert("Please enter all the parameters needed and select all the needed buttons,than
k you :)\nYou can also select a demo case from the demo menu.");

    else{

    Group2->deactivate();
    GroupLanes->deactivate();
    GroupTInterval->deactivate();

    firstCarQ1->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ2->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ3->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ4->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ5->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ6->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ7->deptime=sample_normal(AvgDepT,SDDT);
    firstCarQ8->deptime=sample_normal(AvgDepT,SDDT);


    for(int y=0;y<(runtime/timeInterval);y++)
    {
        Time=(double)clock();

        trafficLight(y);
        increaseCar(y);
        showOut();

        time1->value(timeInterval+time1->value());

        decreaseCar(y);
        Group->redraw();
        window->redraw();
        Fl::check();

        //while((double)clock() <= (Time+100));

    }
```

```
    divloop=y;

    Group2->activate();
    GroupLanes->activate();
    GroupTInterval->activate();

    Grun->deactivate();

    calstat(divloop);
    tablegen();


    tests << "finish "<<MAXq <<endl<<MINq<<endl<<divloop<<endl;;
    tests.close();
    }
}



//////////////////////////////////////////


void trafficLight(int iteration){

    int y=((int)(iteration*timeInterval)/10)%MaxLane;
    ofstream tests("test.txt",ios::app);
    //tests << y <<" y"<<endl;


    if(y != lastY) {
            if(lastY == 66);
            else GreenLDuration+=10;
            lastY=y;


        if(NofLanes == 1){

            for(int x=0;x<4;x++){
                if(TLS[q[x]][y] == 'R') {
                    openLane[q[x]]=0;
                    Tlight[q[x]]->color(1,0);
                    //Tlight[q[x]]->set();
                    Tlight[q[x]]->redraw();


                }
                else if(TLS[q[x]][y] == 'G') {
                    openLane[q[x]]=1;
                    Tlight[q[x]]->color(2,0);
                    //Tlight[q[x]]->set();
                    Tlight[q[x]]->redraw();
                }
                tests << openLane[q[x]];
            }
        }

        else if(NofLanes == 2){
            for(int x=0;x<8;x++){
                if(TLS[x][y] == 'R') {
                    openLane[x]=0;
                    Tlight[x]->color(1,0);
                    //Tlight[x]->set();
                    Tlight[x]->redraw();
                }
                else if(TLS[x][y] == 'G') {
                    openLane[x]=1;
                    Tlight[x]->color(2,0);
                    //Tlight[x]->set();
                    Tlight[x]->redraw();
                }
                tests << openLane[x];
            }
        }



        for(int i=0;i<8;i++){
```

```
            if(LopenLane[i] == openLane[i]){
                if(openLane[i] == 1) gSEQ[i]+=10;}

            else {
                gSEQ[i]=0;
                Gdeptime[i]=0;
            }

            LopenLane[i] = openLane[i];
        }


    }

    tests<<endl;

    tests.close();
}



//////////////////////////////////////////



void increaseCar(int iteration){

    int col=0;

    ofstream tests("test.txt",ios::app);

    int ncars=poisson(arrivalrate),i;

    if(NofLanes == 1){

        for(i=1;i<=ncars;i++){

            currentCarQ1->next= new (struct carT);
            Nq1++;
            currentCarQ1=currentCarQ1->next;
            *currentCarQ1=car;
            currentCarQ1->colorR=rand()%256;
            currentCarQ1->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ1->colorB=col;}
            while(col == 45);
            currentCarQ1->timeR=(iteration * timeInterval);
            currentCarQ1->deptime=sample_normal(AvgDepT,SDDT);

        }ncars=poisson(arrivalrate);increaseStat+=Nq1;outQ[0]->value(Nq1);outQ[0]->redraw();
        tests <<Nq1<<" Cars" <<endl;MAXMINq(Nq1,0);Qlengths[0]+=Nq1;



        for(i=1;i<=ncars;i++){

            currentCarQ3->next= new (struct carT);
            Nq3++;
            currentCarQ3=currentCarQ3->next;
            *currentCarQ3=car;
            currentCarQ3->colorR=rand()%256;
            currentCarQ3->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ3->colorB=col;}
            while(col == 45);
            currentCarQ3->timeR=(iteration * timeInterval);
            currentCarQ3->deptime=sample_normal(AvgDepT,SDDT);

        }ncars=poisson(arrivalrate);increaseStat+=Nq3;outQ[2]->value(Nq3);outQ[2]->redraw();
        tests <<Nq3<<" Cars" <<endl;MAXMINq(Nq3,2);Qlengths[2]+=Nq3;

        for(i=1;i<=ncars;i++){

            currentCarQ5->next= new (struct carT);
            Nq5++;
            currentCarQ5=currentCarQ5->next;
            *currentCarQ5=car;
```

```
            currentCarQ5->colorR=rand()%256;
            currentCarQ5->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ5->colorB=col;}
            while(col == 45);
            currentCarQ5->timeR=(iteration * timeInterval);
            currentCarQ5->deptime=sample_normal(AvgDepT,SDDT);

        }ncars=poisson(arrivalrate);increaseStat+=Nq5;outQ[4]->value(Nq5);outQ[4]->redraw();
        tests <<Nq5<<" Cars" <<endl;MAXMINq(Nq5,4);Qlengths[4]+=Nq5;

        for(i=1;i<=ncars;i++){
            currentCarQ7->next= new (struct carT);
            Nq7++;
            currentCarQ7=currentCarQ7->next;
            *currentCarQ7=car;
            currentCarQ7->colorR=rand()%256;
            currentCarQ7->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ7->colorB=col;}
            while(col == 45);
            currentCarQ7->timeR=(iteration * timeInterval);
            currentCarQ7->deptime=sample_normal(AvgDepT,SDDT);

        }increaseStat+=Nq7;tests <<Nq7<<" Cars" <<endl;outQ[6]->value(Nq7);outQ[6]->redraw();
         MAXMINq(Nq7,6);Qlengths[6]+=Nq7;


//testing
/*

        currentCarQ1=firstCarQ1;

        do{

            tests << currentCarQ1->colorR <<" "<< currentCarQ1->colorG <<" "<< currentCarQ1->co
lorB <<" "<<currentCarQ1->timeR<<endl;
            currentCarQ1=currentCarQ1->next;

        }while( currentCarQ1->next != NULL);
*/
//end of testing

    }

    else if(NofLanes == 2) {

        for(i=1;i<=ncars;i++){

            currentCarQ1->next= new (struct carT);
            Nq1++;
            currentCarQ1=currentCarQ1->next;
            *currentCarQ1=car;
            currentCarQ1->colorR=rand()%256;
            currentCarQ1->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ1->colorB=col;}
            while(col == 45);
            currentCarQ1->timeR=(iteration * timeInterval);
            currentCarQ1->deptime=sample_normal(AvgDepT,SDDT);
        }ncars=poisson(arrivalrate);increaseStat+=Nq1;outQ[0]->value(Nq1);outQ[0]->redraw();
        tests <<Nq1<<" Cars" <<endl;MAXMINq(Nq1,0);Qlengths[0]+=Nq1;

        for(i=1;i<=ncars;i++){

            currentCarQ2->next= new (struct carT);
            Nq2++;
            currentCarQ2=currentCarQ2->next;
            *currentCarQ2=car;
            currentCarQ2->colorR=rand()%256;
            currentCarQ2->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ2->colorB=col;}
            while(col == 45);
            currentCarQ2->timeR=(iteration * timeInterval);
            currentCarQ2->deptime=sample_normal(AvgDepT,SDDT);
        }ncars=poisson(arrivalrate);increaseStat+=Nq2;outQ[1]->value(Nq2);outQ[1]->redraw();
```

```
tests <<Nq2<<" Cars" <<endl;MAXMINq(Nq2,1);Qlengths[1]+=Nq2;

for(i=1;i<=ncars;i++){

    currentCarQ3->next= new (struct carT);
    Nq3++;
    currentCarQ3=currentCarQ3->next;
    *currentCarQ3=car;
    currentCarQ3->colorR=rand()%256;
    currentCarQ3->colorG=rand()%256;
    do{col=(rand()%256);
        currentCarQ3->colorB=col;}
    while(col == 45);
    currentCarQ3->timeR=(iteration * timeInterval);
    currentCarQ3->deptime=sample_normal(AvgDepT,SDDT);
}ncars=poisson(arrivalrate);increaseStat+=Nq3;outQ[2]->value(Nq3);outQ[2]->redraw();
tests <<Nq3<<" Cars" <<endl;MAXMINq(Nq3,2);Qlengths[2]+=Nq3;

for(i=1;i<=ncars;i++){

    currentCarQ4->next= new (struct carT);
    Nq4++;
    currentCarQ4=currentCarQ4->next;
    *currentCarQ4=car;
    currentCarQ4->colorR=rand()%256;
    currentCarQ4->colorG=rand()%256;
    do{col=(rand()%256);
        currentCarQ4->colorB=col;}
    while(col == 45);
    currentCarQ4->timeR=(iteration * timeInterval);
    currentCarQ4->deptime=sample_normal(AvgDepT,SDDT);
}ncars=poisson(arrivalrate);increaseStat+=Nq4;outQ[3]->value(Nq4);outQ[3]->redraw();
tests <<Nq4<<" Cars" <<endl;MAXMINq(Nq4,3);Qlengths[3]+=Nq4;

for(i=1;i<=ncars;i++){

    currentCarQ5->next= new (struct carT);
    Nq5++;
    currentCarQ5=currentCarQ5->next;
    *currentCarQ5=car;
    currentCarQ5->colorR=rand()%256;
    currentCarQ5->colorG=rand()%256;
    do{col=(rand()%256);
        currentCarQ5->colorB=col;}
    while(col == 45);
    currentCarQ5->timeR=(iteration * timeInterval);
    currentCarQ5->deptime=sample_normal(AvgDepT,SDDT);
}ncars=poisson(arrivalrate);increaseStat+=Nq5;outQ[4]->value(Nq5);outQ[4]->redraw();
tests <<Nq5<<" Cars" <<endl;MAXMINq(Nq5,4);Qlengths[4]+=Nq5;

for(i=1;i<=ncars;i++){

    currentCarQ6->next= new (struct carT);
    Nq6++;
    currentCarQ6=currentCarQ6->next;
    *currentCarQ6=car;
    currentCarQ6->colorR=rand()%256;
    currentCarQ6->colorG=rand()%256;
    do{col=(rand()%256);
        currentCarQ6->colorB=col;}
    while(col == 45);
    currentCarQ6->timeR=(iteration * timeInterval);
    currentCarQ6->deptime=sample_normal(AvgDepT,SDDT);
}ncars=poisson(arrivalrate);increaseStat+=Nq6;outQ[5]->value(Nq6);outQ[5]->redraw();
tests <<Nq6<<" Cars" <<endl;MAXMINq(Nq6,5);Qlengths[5]+=Nq6;

for(i=1;i<=ncars;i++){

    currentCarQ7->next= new (struct carT);
    Nq7++;
    currentCarQ7=currentCarQ7->next;
    *currentCarQ7=car;
    currentCarQ7->colorR=rand()%256;
    currentCarQ7->colorG=rand()%256;
    do{col=(rand()%256);
        currentCarQ7->colorB=col;}
    while(col == 45);
```

```
            currentCarQ7->timeR=(iteration * timeInterval);
            currentCarQ7->deptime=sample_normal(AvgDepT,SDDT);
        }ncars=poisson(arrivalrate);increaseStat+=Nq7;outQ[6]->value(Nq7);outQ[6]->redraw();
        tests <<Nq7<<" Cars" <<endl;MAXMINq(Nq7,6);Qlengths[6]+=Nq7;

        for(i=1;i<=ncars;i++){

            currentCarQ8->next= new (struct carT);
            Nq8++;
            currentCarQ8=currentCarQ8->next;
            *currentCarQ8=car;
            currentCarQ8->colorR=rand()%256;
            currentCarQ8->colorG=rand()%256;
            do{col=(rand()%256);
                currentCarQ8->colorB=col;}
            while(col == 45);
            currentCarQ8->timeR=(iteration * timeInterval);
            currentCarQ8->deptime=sample_normal(AvgDepT,SDDT);
        }increaseStat+=Nq8;outQ[7]->value(Nq8);outQ[7]->redraw();
        tests <<Nq8<<" Cars" <<endl;MAXMINq(Nq8,7);Qlengths[7]+=Nq8;


    }



    tests.close();
}


////////////////////////////////////////////


void decreaseCar(int iteration)
{

    ofstream tests("test.txt",ios::app);
    double Gtime[8];

    for(int i=0;i<8;i++){
        Gtime[i]=(((iteration+1)*timeInterval)-GreenLDuration)+gSEQ[i]-Gdeptime[i];
    }
    carT *p;


    if(NofLanes == 1)
    {
        for(int x=0;x<4;x++)
        {

            tests<<Gtime[q[x]]<<" Duration of last green light" <<endl;

            if(openLane[q[x]] == 0) continue;

            else
            {


                while(1){

                        if(q[x] == 0){
                            if(Nq1 == 0) break;
                            if((firstCarQ1->deptime) < Gtime[q[x]]){
                                tests <<"Car removed q1"<<endl;
                                Gtime[q[x]]-=(firstCarQ1->deptime);
                                Gdeptime[q[x]]+=firstCarQ1->deptime;
                                Qwt[q[x]]+=firstCarQ1->deptime;
                                tests <<"Car deptime: "<<(firstCarQ1->deptime)<<" Time Left: "<
<Gtime[q[x]]<<endl;
                                Nq1--;numbofCars[q[x]]++;
                                p = firstCarQ1;
                                decreseStats=((iteration)*timeInterval)-firstCarQ1->timeR;
                                MAXMINwt (decreseStats,0);
                                firstCarQ1 = firstCarQ1->next;
                                delete p;
                            }
                            else break;
```

```
                        }

                        if(q[x] == 2){
                            if(Nq3 == 0) break;
                            if((firstCarQ3->deptime) < Gtime[q[x]]){
                                tests <<"Car removed q3"<<endl;
                                Gtime[q[x]]-=(firstCarQ3->deptime);
                                Gdeptime[q[x]]+=firstCarQ3->deptime;
                                Qwt[q[x]]+=firstCarQ3->deptime;
                                tests <<"Car deptime: "<<(firstCarQ3->deptime)<<" Time Left: "<
<Gtime[q[x]]<<endl;
                                Nq3--;numbofCars[q[x]]++;
                                p = firstCarQ3;
                                decreseStats=((iteration)*timeInterval)-firstCarQ3->timeR;
                                MAXMINwt (decreseStats,2);
                                firstCarQ3 = firstCarQ3->next;
                                delete p;
                            }
                            else break;
                        }

                        if(q[x] == 4){
                            if(Nq5 == 0) break;
                            if((firstCarQ5->deptime) < Gtime[q[x]]){
                                tests <<"Car removed q5"<<endl;
                                Gtime[q[x]]-=(firstCarQ5->deptime);
                                Gdeptime[q[x]]+=firstCarQ5->deptime;
                                Qwt[q[x]]+=firstCarQ5->deptime;
                                tests <<"Car deptime: "<<(firstCarQ5->deptime)<<" Time Left: "<
<Gtime[q[x]]<<endl;
                                Nq5--;numbofCars[q[x]]++;
                                p = firstCarQ5;
                                decreseStats=((iteration)*timeInterval)-firstCarQ5->timeR;
                                MAXMINwt (decreseStats,4);
                                firstCarQ5 = firstCarQ5->next;
                                delete p;
                            }
                            else break;
                        }

                        if(q[x] == 6){
                            if(Nq7 == 0) break;
                            if((firstCarQ7->deptime) < Gtime[q[x]]){
                                tests <<"Car removed q7"<<endl;
                                Gtime[q[x]]-=(firstCarQ7->deptime);
                                Gdeptime[q[x]]+=firstCarQ7->deptime;
                                Qwt[q[x]]+=firstCarQ7->deptime;
                                tests <<"Car deptime: "<<(firstCarQ7->deptime)<<" Time Left: "<
<Gtime[q[x]]<<endl;
                                Nq7--;numbofCars[q[x]]++;
                                p = firstCarQ7;
                                decreseStats=((iteration)*timeInterval)-firstCarQ7->timeR;
                                MAXMINwt (decreseStats,6);
                                firstCarQ7 = firstCarQ7->next;
                                delete p;
                            }
                            else break;
                        }
                    }

            }
        }
    }

    //tests <<(((iteration+1)*timeInterval)-GreenLDuration)<<" HERE"<<endl;
    //tests.close();



    else if(NofLanes == 2)
    {

        for(int x=0;x<8;x++)
        {
            tests<<Gtime[x]<<" Duration of last green light" <<endl;
```

```
            if(openLane[x] == 0) continue;

            else
            {
                    while(1)
                    {

                        if(x == 0){
                            if(Nq1 == 0) break;
                            if((firstCarQ1->deptime) < Gtime[x]){
                                tests <<"Car removed q1"<<endl;
                                Gtime[x]-=(firstCarQ1->deptime);
                                Gdeptime[x]+=firstCarQ1->deptime;
                                Qwt[x]+=firstCarQ1->deptime;
                                tests <<"Car deptime: "<<(firstCarQ1->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                                Nq1--;numbofCars[x]++;
                                p = firstCarQ1;
                                decreseStats=((iteration)*timeInterval)-firstCarQ1->timeR;

                                MAXMINwt (decreseStats,0);
                                firstCarQ1 = firstCarQ1->next;
                                delete p;
                            }
                            else break;
                        }

                        if(x == 1){
                            if(Nq2 == 0) break;
                            if((firstCarQ2->deptime) < Gtime[x]){
                                tests <<"Car removed q2"<<endl;
                                Gtime[x]-=(firstCarQ2->deptime);
                                Gdeptime[x]+=firstCarQ2->deptime;
                                Qwt[x]+=firstCarQ2->deptime;
                                tests <<"Car deptime: "<<(firstCarQ2->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                                Nq2--;numbofCars[x]++;
                                p = firstCarQ2;
                                decreseStats=((iteration)*timeInterval)-firstCarQ2->timeR;
                                MAXMINwt (decreseStats,1);
                                firstCarQ2 = firstCarQ2->next;
                                delete p;
                            }
                            else break;
                        }

                        if(x == 2){
                            if(Nq3 == 0) break;
                            if((firstCarQ3->deptime) < Gtime[x]){
                                tests <<"Car removed q3"<<endl;
                                Gtime[x]-=(firstCarQ3->deptime);
                                Gdeptime[x]+=firstCarQ3->deptime;
                                Qwt[x]+=firstCarQ3->deptime;
                                tests <<"Car deptime: "<<(firstCarQ3->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                                Nq3--;numbofCars[x]++;
                                p = firstCarQ3;
                                decreseStats=((iteration)*timeInterval)-firstCarQ3->timeR;
                                MAXMINwt (decreseStats,2);
                                firstCarQ3 = firstCarQ3->next;
                                delete p;
                            }
                            else break;
                        }

                        if(x == 3){
                            if(Nq4 == 0) break;
                            if((firstCarQ4->deptime) < Gtime[x]){
                                tests <<"Car removed q4"<<endl;
                                Gtime[x]-=(firstCarQ4->deptime);
                                Gdeptime[x]+=firstCarQ4->deptime;
                                Qwt[x]+=firstCarQ4->deptime;
                                tests <<"Car deptime: "<<(firstCarQ4->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                                Nq4--;numbofCars[x]++;
                                p = firstCarQ4;
                                decreseStats=((iteration)*timeInterval)-firstCarQ4->timeR;
```

```
                        MAXMINwt (decreseStats,3);
                        firstCarQ4 = firstCarQ4->next;
                        delete p;
                    }
                    else break;
                }

                if(x == 4){
                    if(Nq5 == 0) break;
                    if((firstCarQ5->deptime) < Gtime[x]){
                        tests <<"Car removed q5"<<endl;
                        Gtime[x]-=(firstCarQ5->deptime);
                        Gdeptime[x]+=firstCarQ5->deptime;
                        Qwt[x]+=firstCarQ5->deptime;
                        tests <<"Car deptime: "<<(firstCarQ1->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                        Nq5--;numbofCars[x]++;
                        p = firstCarQ5;
                        decreseStats=((iteration)*timeInterval)-firstCarQ5->timeR;
                        MAXMINwt (decreseStats,4);
                        firstCarQ5 = firstCarQ5->next;
                        delete p;
                    }
                    else break;
                }

                if(x == 5){
                    if(Nq6 == 0) break;
                    if((firstCarQ6->deptime) < Gtime[x]){
                        tests <<"Car removed q6"<<endl;
                        Gtime[x]-=(firstCarQ6->deptime);
                        Gdeptime[x]+=firstCarQ6->deptime;
                        Qwt[x]+=firstCarQ6->deptime;
                        tests <<"Car deptime: "<<(firstCarQ6->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                        Nq6--;numbofCars[x]++;
                        p = firstCarQ6;
                        decreseStats=((iteration)*timeInterval)-firstCarQ6->timeR;
                        MAXMINwt (decreseStats,5);
                        firstCarQ6 = firstCarQ6->next;
                        delete p;
                    }
                    else break;
                }

                if(x == 6){
                    if(Nq7 == 0) break;
                    if((firstCarQ7->deptime) < Gtime[x]){
                        tests <<"Car removed q7"<<endl;
                        Gtime[x]-=(firstCarQ7->deptime);
                        Gdeptime[x]+=firstCarQ7->deptime;
                        Qwt[x]+=firstCarQ7->deptime;
                        tests <<"Car deptime: "<<(firstCarQ7->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                        Nq7--;numbofCars[x]++;
                        p = firstCarQ7;
                        decreseStats=((iteration)*timeInterval)-firstCarQ7->timeR;
                        MAXMINwt (decreseStats,6);
                        firstCarQ7 = firstCarQ7->next;
                        delete p;
                    }
                    else break;
                }

                if(x == 7){
                    if(Nq8 == 0) break;
                    if((firstCarQ8->deptime) < Gtime[x]){
                        tests <<"Car removed q8"<<endl;
                        Gtime[x]-=(firstCarQ8->deptime);
                        Gdeptime[x]+=firstCarQ8->deptime;
                        Qwt[x]+=firstCarQ8->deptime;
                        tests <<"Car deptime: "<<(firstCarQ8->deptime)<<" Time Left: "<
<Gtime[x]<<endl;
                        Nq8--;numbofCars[x]++;
                        p = firstCarQ8;
                        decreseStats=((iteration)*timeInterval)-firstCarQ8->timeR;
                        MAXMINwt (decreseStats,7);
```

```
                                firstCarQ8 = firstCarQ8->next;
                                delete p;
                            }
                            else break;
                        }
                    }

                }
            }
        }
```

```
//////////////////////////////////////////////


void showOut(){

    int cars=0;
    Fl_Box *del;

    for(int y=0;y<8;y++){

        if(drawn[y]>0)
        {

            for(int i=0;i<drawn[y];i++)
            {
                Group->remove(test[y][i]);
                delete test[y][i];
                test[y][i]= NULL;
            }
        }

    }

        drawn[0]=0;
        if(Nq1 > 15) cars=15;
        else cars=Nq1;
        q1=firstCarQ1;

        for(int i=0;i<cars;i++){

                        test[0][i]=new Fl_Box(197,(160-(6*i)),5,5);
                        test[0][i]->box(FL_FLAT_BOX);
                        //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                        test[0][i]->color(q1->colorB);
                        test[0][i]->redraw();
                        Group->add(test[0][i]);
                        drawn[0]++;
                        if(q1->next == NULL);
                        else q1=q1->next;
                    }


        drawn[2]=0;
        if(Nq3 > 15) cars=15;
        else cars=Nq3;
        q1=firstCarQ3;

        for(i=0;i<cars;i++){

                        test[2][i]=new Fl_Box((285+(6*i)),205,5,5);
                        test[2][i]->box(FL_FLAT_BOX);
                        //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                        test[2][i]->color(q1->colorB);
                        test[2][i]->redraw();
                        Group->add(test[2][i]);
                        drawn[2]++;
                        if(q1->next == NULL);
                        else q1=q1->next;
                    }
```

```
drawn[4]=0;
if(Nq5 > 15) cars=15;
else cars=Nq5;
q1=firstCarQ5;

for(i=0;i<cars;i++){

                test[4][i]=new Fl_Box(250,(295+(6*i)),5,5);
                test[4][i]->box(FL_FLAT_BOX);
                //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                test[4][i]->color(q1->colorB);
                test[4][i]->redraw();
                Group->add(test[4][i]);
                drawn[4]++;
                if(q1->next == NULL);
                else q1=q1->next;
            }


drawn[6]=0;
if(Nq7 > 15) cars=15;
else cars=Nq7;
q1=firstCarQ7;

for(i=0;i<cars;i++){

                test[6][i]=new Fl_Box((150-(6*i)),260,5,5);
                test[6][i]->box(FL_FLAT_BOX);
                //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                test[6][i]->color(q1->colorB);
                test[6][i]->redraw();
                Group->add(test[6][i]);
                drawn[6]++;
                if(q1->next == NULL);
                else q1=q1->next;
            }






if(NofLanes == 2){

    drawn[1]=0;
    if(Nq2 > 15) cars=15;
    else cars=Nq2;
    q1=firstCarQ2;

    for(i=0;i<cars;i++){

                    test[1][i]=new Fl_Box(217,(160-(6*i)),5,5);
                    test[1][i]->box(FL_FLAT_BOX);
                    //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                    test[1][i]->color(q1->colorB);
                    test[1][i]->redraw();
                    Group->add(test[1][i]);
                    drawn[1]++;
                    if(q1->next == NULL);
                    else q1=q1->next;
                }


    drawn[3]=0;
    if(Nq4 > 15) cars=15;
    else cars=Nq4;
    q1=firstCarQ4;

    for(i=0;i<cars;i++){

                    test[3][i]=new Fl_Box((285+(6*i)),223,5,5);
                    test[3][i]->box(FL_FLAT_BOX);
                    //c1=fl_rgb_color(,q1->colorG,q1->colorB);
                    test[3][i]->color(q1->colorB);
                    test[3][i]->redraw();
                    Group->add(test[3][i]);
```

```
                            drawn[3]++;
                            if(q1->next == NULL);
                            else q1=q1->next;
                        }


            drawn[5]=0;
            if(Nq6 > 15) cars=15;
            else cars=Nq6;
            q1=firstCarQ6;

            for(i=0;i<cars;i++){

                            test[5][i]=new Fl_Box(233,(295+(6*i)),5,5);
                            test[5][i]->box(FL_FLAT_BOX);
                            test[5][i]->color(q1->colorB);
                            test[5][i]->redraw();
                            Group->add(test[5][i]);
                            drawn[5]++;
                            if(q1->next == NULL);
                            else q1=q1->next;
                        }

            drawn[7]=0;
            if(Nq8 > 15) cars=15;
            else cars=Nq8;
            q1=firstCarQ8;

            for(i=0;i<cars;i++){

                            test[7][i]=new Fl_Box((150-(6*i)),242,5,5);
                            test[7][i]->box(FL_FLAT_BOX);
                            test[7][i]->color(q1->colorB);
                            test[7][i]->redraw();
                            Group->add(test[7][i]);
                            drawn[7]++;
                            if(q1->next == NULL);
                            else q1=q1->next;
                        }

        }




}


////////////////////////////////////////////


double sample_normal(double mean, double std_dev) {

    // generate a random sample from a Normal distribution with a given mean and standard devia
tion
    // use the function rand to generate a random number

    double c, z=0, w, w1, w2, urs; // urs: uniform random sample

    // now compute the random sample
    while(z <= 0){
        do {
            urs = rand() / (double) RAND_MAX; // generate a uniform random sample in the interv
al 0-1
            w1= 2*urs -1;

            urs = rand() / (double) RAND_MAX;
            w2= 2*urs -1;

            w = w1*w1 + w2*w2;
        } while (w >= 1);

        c = sqrt((-2*log(w)) / w);

        z = c * w1;               // z is a random sample for standard normal distribution
                                  // i.e. with mean of zero and variance 1
                                  // we could also have assigned z = c / w2;
```

```
        z = mean + z * std_dev;  // convert it to the required normal distribution
    }

    return (z);

}



////////////////////////////////////////////////////////


void MAXMINq(int qlength,int i){

    if(qlength > MAXq[i])
        MAXq[i]=qlength;

    if(qlength < MINq[i] && qlength != 0)
        MINq[i]=qlength;
}
////////////////////////////////////////////////////////

void MAXMINwt (int wtime,int i){

    if(wtime > MAXwt[i])
        MAXwt[i]=wtime;

    if(wtime < MINwt[i])
        MINwt[i]=wtime;

}

////////////////////////////////////////////////////////

void reinitCB(Fl_Widget *w,void *v){

    carT *p;
    int col;

    for(int lo=0;lo<Nq1;lo++){
        if(Nq1 == 0) break;
        p=firstCarQ1;
        firstCarQ1=firstCarQ1->next;
        delete p;
    }

    if(firstCarQ1 != NULL) delete firstCarQ1;
    firstCarQ1=new (struct carT);
    *firstCarQ1= car;
    currentCarQ1=firstCarQ1;
    do{col=(rand()%256);
       firstCarQ1->colorB=col;}
    while(col == 45);



    for( lo=0;lo<Nq2;lo++){
        if(Nq2 == 0) break;
        p=firstCarQ2;
        firstCarQ2=firstCarQ2->next;
        delete p;
    }

    if(firstCarQ2 != NULL) delete firstCarQ2;
    firstCarQ2=new (struct carT);
    *firstCarQ2= car;
    currentCarQ2=firstCarQ2;
    do{col=(rand()%256);
       firstCarQ2->colorB=col;}
    while(col == 45);



    for( lo=0;lo<Nq3;lo++){
        if(Nq3 == 0) break;
        p=firstCarQ3;
```

```
    firstCarQ3=firstCarQ3->next;
    delete p;
}

if(firstCarQ3 != NULL) delete firstCarQ3;
firstCarQ3=new (struct carT);
*firstCarQ3= car;
currentCarQ3=firstCarQ3;
do{col=(rand()%256);
   firstCarQ3->colorB=col;}
while(col == 45);



for( lo=0;lo<Nq4;lo++){
    if(Nq4 == 0) break;
    p=firstCarQ4;
    firstCarQ4=firstCarQ4->next;
    delete p;
}

if(firstCarQ4 != NULL) delete firstCarQ4;
firstCarQ4=new (struct carT);
*firstCarQ4= car;
currentCarQ4=firstCarQ4;
do{col=(rand()%256);
   firstCarQ4->colorB=col;}
while(col ==45);



for( lo=0;lo<Nq5;lo++){
    if(Nq5 == 0) break;
    p=firstCarQ5;
    firstCarQ5=firstCarQ5->next;
    delete p;
}

if(firstCarQ5 != NULL) delete firstCarQ5;
firstCarQ5=new (struct carT);
*firstCarQ5= car;
currentCarQ5=firstCarQ5;
do{col=(rand()%256);
   firstCarQ5->colorB=col;}
while(col == 45);



for( lo=0;lo<Nq6;lo++){
    if(Nq6 == 0) break;
    p=firstCarQ6;
    firstCarQ6=firstCarQ6->next;
    delete p;
}

if(firstCarQ6 != NULL) delete firstCarQ6;
firstCarQ6=new (struct carT);
*firstCarQ6= car;
currentCarQ6=firstCarQ6;
do{col=(rand()%256);
   firstCarQ6->colorB=col;}
while(col == 45);



for( lo=0;lo<Nq7;lo++){
    if(Nq7 == 0) break;
    p=firstCarQ7;
    firstCarQ7=firstCarQ7->next;
    delete p;
}

if(firstCarQ7 != NULL) delete firstCarQ7;
firstCarQ7=new (struct carT);
*firstCarQ7= car;
currentCarQ7=firstCarQ7;
do{col=(rand()%256);
```

```
   firstCarQ7->colorB=col;}
while(col == 45);



for( lo=0;lo<Nq8;lo++){
    if(Nq8 == 0) break;
    p=firstCarQ8;
    firstCarQ8=firstCarQ8->next;
    delete p;
}

if(firstCarQ8 != NULL) delete firstCarQ8;
firstCarQ8=new (struct carT);
*firstCarQ8= car;
currentCarQ8=firstCarQ8;
do{col=(rand()%256);
   firstCarQ8->colorB=col;}
while(col == 45);


onesecinterval->value(0);
tensecinterval->value(0);
onemininterval->value(0);

onelane->value(0);
twolane->value(0);

arrivalinput->value(0);
runningtimeinput->value(0);
avgdepinput->value(0);

Group2->redraw();

for(lo=0;lo<8;lo++){
    outQ[lo]->value(0);
    outQ[lo]->redraw();
    outQ[lo]->hide();
    Tlight[lo]->color(45,0);
    Tlight[lo]->hide();
}

time1->value(0);

arrivalrate=-1;
runtime=0;
NofLanes=0;
increaseStat=0;
GreenLDuration=0;
lastY=66;
decreseStats=0;
draw=0;
divloop=0;


Nq1=0;
Nq2=0;
Nq3=0;
Nq4=0;
Nq5=0;
Nq6=0;
Nq7=0;
Nq8=0;

AvgDepT=-1.0;
timeInterval=0.0;


for(int y=0;y<8;y++){

    if(drawn[y]>0)
    {

        for(int i=0;i<drawn[y];i++)
        {
            Group->remove(test[y][i]);
            delete test[y][i];
```

```
                test[y][i]= NULL;
            }
        }

    }

    Group->redraw();
    window->redraw();

    for(int i=0;i<8;i++){

        openLane[i]=0;
        gSEQ[i]=0;
        LopenLane[i]=0;
        drawn[i]=0;
        numbofCars[i]=0;
        Qlengths[i]=0;
        Qavg[i]=0;


        Qwt[i]=0;
        Gdeptime[i]=0;
        Wavg[i]=0;
        MAXq[i]=0;
        MINq[i]=35345;
        MAXwt[i]=0;
        MINwt[i]=2222;
    }

    Grun->activate();

}
//////////////////////////////////////////////////////
void calstat(int loops){

ofstream tests("test.txt",ios::app);

    for(int i=0;i<8;i++){
        Qavg[i]=(double)Qlengths[i]/(double)loops;
        Wavg[i]=Qwt[i]/numbofCars[i];
        tests<<"QAvg for Q"<<i<<":"<<Qavg[i]<<"  "<<"WAvg for Q"<<i<<":"<<Wavg[i]<<endl;

    }

    for( i=0;i<8;i++){
        tests<<endl<<"Qlngth for Q"<<i<<":"<<Qlengths[i]<<"  "<<"Wt for Q"<<i<<":"<<Qwt[i]<<"
"<<"Cars for Q"<<i<<":"<<numbofCars[i]<<endl;

    }
}

//////////////////////////////////////////////////////////
void tablegen(){

    double temp1,temp2,temp3,temp4,temp5,temp6;


        temp1=(Qavg[0]+Qavg[1]+Qavg[2]+Qavg[3]+Qavg[4]+Qavg[5]+Qavg[6]+Qavg[7])/(NofLanes== 1?4
:8);
        temp2=(MAXq[0]+MAXq[1]+MAXq[2]+MAXq[3]+MAXq[4]+MAXq[5]+MAXq[6]+MAXq[7])/(NofLanes== 1?4
:8);
        temp3=(MINq[0]+MINq[1]+MINq[2]+MINq[3]+MINq[4]+MINq[5]+MINq[6]+MINq[7])/(NofLanes== 1?4
:8);
        temp4=(Qwt[0]+Qwt[1]+Qwt[2]+Qwt[3]+Qwt[4]+Qwt[5]+Qwt[6]+Qwt[7])/(NofLanes== 1?4:8);
        temp5=(MAXwt[0]+MAXwt[1]+MAXwt[2]+MAXwt[3]+MAXwt[4]+MAXwt[5]+MAXwt[6]+MAXwt[7])/(NofLan
es== 1?4:8);
        temp6=(MINwt[0]+MINwt[1]+MINwt[2]+MINwt[3]+MINwt[4]+MINwt[5]+MINwt[6]+MINwt[7])/(NofLan
es== 1?4:8);

    ofstream tests("results.xls",ios::out);

    if(NofLanes==1)  {


      tests<<"\t\t\t\t\tQueue 1\tQueue 2\tQueue 3\tQueue 4\tAverage\t"<<endl;
      tests<<"\t\tAverage Queue Length(cars)\t\t\t"<<Qavg[0]<<"\t"<<Qavg[2]<<"\t"<<Qavg[4]<<"\t
"<<Qavg[6]<<"\t"<<temp1<<endl;
```

```
      tests<<"\t\tMax Queue Length(cars)\t\t\t"<<MAXq[0]<<"\t"<<MAXq[2]<<"\t"<<MAXq[4]<<"\t"<<M
AXq[6]<<"\t"<<temp2<<endl;
      tests<<"\t\tMin Queue Length(cars)\t\t\t"<<MINq[0]<<"\t"<<MINq[2]<<"\t"<<MINq[4]<<"\t"<<M
INq[6]<<"\t"<<temp3<<endl;
      tests<<endl;
      tests<<"\t\tAverage Waiting Time(sec)\t\t\t"<<Qwt[0]<<"\t"<<Qwt[2]<<"\t"<<Qwt[4]<<"\t"<<Q
wt[6]<<"\t"<<temp4<<endl;
      tests<<"\t\tMax Waiting Time(sec)\t\t\t"<<MAXwt[0]<<"\t"<<MAXwt[2]<<"\t"<<MAXwt[4]<<"\t"<
<MAXwt[6]<<"\t"<<temp5<<endl;
      tests<<"\t\tMin Waiting Time(sec)\t\t\t"<<MINwt[0]<<"\t"<<MINwt[2]<<"\t"<<MINwt[4]<<"\t"<
<MINwt[6]<<"\t"<<temp6<<endl;

                  }

      else if(NofLanes==2){

      tests<<"\t\t\t\t\tQueue 1\tQueue 2\tQueue 3\tQueue 4\tQueue 5\tQueue 6\tQueue 7\tQueue 8\
tAverage\t"<<endl;
      tests<<"\t\tAverage Queue Length(cars)\t\t\t"<<Qavg[0]<<"\t"<<Qavg[1]<<"\t"<<Qavg[2]<<"\t
"<<Qavg[3]<<"\t"<<Qavg[4]<<"\t"<<Qavg[5]<<"\t"<<Qavg[6]<<"\t"<<Qavg[7]<<"\t"<<temp1<<endl;
      tests<<"\t\tMax Queue Length(cars)\t\t\t"<<MAXq[0]<<"\t"<<MAXq[1]<<"\t"<<MAXq[2]<<"\t"<<M
AXq[3]<<"\t"<<MAXq[4]<<"\t"<<MAXq[5]<<"\t"<<MAXq[6]<<"\t"<<MAXq[7]<<"\t"<<temp2<<endl;
      tests<<"\t\tMin Queue Length(cars)\t\t\t"<<MINq[0]<<"\t"<<MINq[1]<<"\t"<<MINq[2]<<"\t"<<M
INq[3]<<"\t"<<MINq[4]<<"\t"<<MINq[5]<<"\t"<<MINq[6]<<"\t"<<MINq[7]<<"\t"<<temp3<<endl;
      tests<<endl;
      tests<<"\t\tAverage Waiting Time(sec)\t\t\t"<<Qwt[0]<<"\t"<<Qwt[1]<<"\t"<<Qwt[2]<<"\t"<<Q
wt[3]<<"\t"<<Qwt[4]<<"\t"<<Qwt[5]<<"\t"<<Qwt[6]<<"\t"<<Qwt[7]<<"\t"<<temp4<<endl;
      tests<<"\t\tMax Waiting Time(sec)\t\t\t"<<MAXwt[0]<<"\t"<<MAXwt[1]<<"\t"<<MAXwt[2]<<"\t"<
<MAXwt[3]<<"\t"<<MAXwt[4]<<"\t"<<MAXwt[5]<<"\t"<<MAXwt[6]<<"\t"<<MAXwt[7]<<"\t"<<temp5<<endl;
      tests<<"\t\tMin Waiting Time(sec)\t\t\t"<<MINwt[0]<<"\t"<<MINwt[1]<<"\t"<<MINwt[2]<<"\t"<
<MINwt[3]<<"\t"<<MINwt[4]<<"\t"<<MINwt[5]<<"\t"<<MINwt[6]<<"\t"<<MINwt[7]<<"\t"<<temp6<<endl;

                  }



    tests.close();

}

////////////////////////////////////////////////////////
void demobut(Fl_Widget *w, void *v){

    if((int) v == 1){

          arrivalinput->value("25");
          arrivalrate = 25;

          avgdepinput->value("2");
          AvgDepT =2;

          runningtimeinput->value("1");
          runtime = 60 ;

          twolane->setonly();
          getuserinputRadio(w,(void*)5);

          onemininterval->setonly();
          timeInterval = 0.1;


    }

    if((int) v == 2){

        arrivalinput->value("25");
          arrivalrate = 25;

          avgdepinput->value("2");
          AvgDepT =2;

          runningtimeinput->value("1");
          runtime = 60 ;

          twolane->setonly();
          getuserinputRadio(w,(void*)4);
```

```
            onemininterval->setonly();
            timeInterval = 0.1;

            }

    if((int) v == 3){

        arrivalinput->value("2500");
            arrivalrate = 2500;

            avgdepinput->value("1");
            AvgDepT =1;

            runningtimeinput->value("1");
            runtime = 60 ;

            twolane->setonly();
            getuserinputRadio(w,(void*)5);

            onemininterval->setonly();
            timeInterval = 0.1;

            }
}
```