Lab 4 - lab.h

```
1  # include <string>
2  # include <cairo.h>
3  # include <config.h>
4  # include <FL/Fl_Cairo_Window.H>
5  # include <car.h>
6  # include <person.h>
7  # include <drive.h>
8  # include <llqueue.h>
9  # include <rbqueue.h>
10
11 using namespace std;
```

- Libraries:
  - string: std::string
  - cairo, config, FL/Fl_Cairo_Window: cairo graphic library
  - car: car struct
  - person: person struct
  - drive: drive struct for moving animation
  - llqueue: linked list queue class
  - rbqueue: ring buffer queue class

Lab 4 - lab.h

```
1 const string    CAR_PATH = "cars/car";
2 const string    PERSON_PATH = "persons/person";
3 const string    PNG = ".png";
4 const int       WIDTH = 600;
5 const int       HEIGHT = 400;
6 const int       MAX_CAR = 5;
7 const double    DEST_X = WIDTH;
8 const double    DEST_Y = HEIGHT * 0.7;
```

- Global const variables:
  - CAR_PATH, PERSON_PATH:
    path to the directories
  - PNG: png file format extension
  - WIDTH, HEIGHT: the sizes for the cairo window
  - MAX_CAR: the number of cars
  - DEST_X, DEST_Y: x,y of destination (driving)

Lab 4 - lab.h

```
1 void    load_cars(void);
2 void    draw_cb(Fl_Cairo_Window *cw, cairo_t *cr);
3 void    draw_background(cairo_t *cr);
4 void    load_drive(void);
5 void    draw_drive(cairo_t *cr);
6 void    arrive_cb(void *);
7 void    redraw_cb(void *);
```

- Function prototypes

Lab 4 - lab.h

```
1 extern LLQUEUE           cars;
2 extern RBQUEUE           people;
3 extern DRIVE             drive;
4 extern Fl_Cairo_Window   cw;
5
6 #endif
```

- Extern variables

Lab 4 - car.h

```
1  #ifndef CAR_H
2  # define CAR_H
3
4  # include <lab.h>
5
6  struct                Car
7  {
8      double            x;
9      double            y;
10     cairo_surface_t *s;
11 };
12
13 #endif
```

- struct Car:
  - it holds the information of cars
  - x,y positions and the image data

Lab 4 - person.h

```
1  #ifndef PERSON_H
2  # define PERSON_H
3
4  # include <lab.h>
5
6  struct                Person
7  {
8      double            x;
9      double            y;
10     cairo_surface_t *s;
11 };
12
13 #endif
```

- struct Person:
  - it holds the information of people
  - x,y positions and the image data

Lab 4 - drive.h

```
1  #ifndef DRIVE_H
2  # define DRIVE_H
3
4  # include <lab.h>
5
6  struct  DRIVE
7  {
8      DRIVE(void) {
9          move = false;
10     };
11     Car      car;
12     Person   person;
13     double   st_x;
14     double   st_y;
15     double   x;
16     double   y;
17     bool     move;
18 };
```

- struct DRIVE:
  - it takes struct Car and Person
  - and holds the information of driving animation

```
1 struct        NODE
2 {
3     Car       car;
4     NODE      *next;
5 };
```

- struct NODE:
  - the node for the linked list

Lab 4 - llqueue.h

```
1  class    LLQUEUE
2  {
3      private:
4          NODE     *front;
5          NODE     *rear;
6
7      public:
8          LLQUEUE();
9          ~LLQUEUE();
10
11         bool     Insert(Car &car);
12         bool     Remove(Car &car);
13         bool     isEmpty();
14         void     draw_cars(cairo_t* cr);
15 };
```

- class LLQUEUE:
  - linked list queue for cars

Lab 4 - llqueue.cpp

```
1  LLQUEUE::LLQUEUE(void)
2  {
3      front = rear = NULL;
4  }
5
6  LLQUEUE::~LLQUEUE(void)
7  {
8      NODE    *tmp;
9
10     while (front)
11     {
12         tmp = front;
13         front = front->next;
14         cairo_surface_destroy(tmp->car.s);
15         delete tmp;
16     }
17 }
```

- member functions of class LLQUEUE
- constructor and destructor
  - destructor frees out all the allocated nodes

Lab 4 - llqueue.cpp

```
1  bool    LLQUEUE::Insert(Car &car)
2  {
3      NODE    *new_node = new NODE;
4
5      if (!new_node)
6          return false;
7      new_node->car = car;
8      new_node->next = NULL;
9      if (rear == 0)
10         front = rear = new_node;
11     else
12     {
13         rear->next = new_node;
14         rear = rear->next;
15     }
16     return true;
17 }
```

- Insert
  - it takes a new car object and puts in a newly allocated node
  - Then, adds the node to the linked list

Lab 4 - llqueue.cpp

```
1  bool    LLQUEUE::Remove(Car &car)
2  {
3      NODE    *tmp;
4
5      if (front == 0)
6          return false;
7      car = front->car;
8      tmp = front;
9      front = front->next;
10     delete tmp;
11     if (front == 0)
12         rear = 0;
13     return true;
14 }
```

- Remove
  - it removes the front node from the linked list and frees out the memory
  - Returns the item of the node

Lab 4 - llqueue.cpp

```
1 bool     LLQUEUE::isEmpty(void)
2 {
3     return (front == NULL);
4 }
```

- isEmpty
  - it checks if the list is empty or not

Lab 4 - llqueue.cpp

```
1 void     LLQUEUE::draw_cars(cairo_t* cr)
2 {
3     for(NODE* n = front; n; n = n->next)
4     {
5         cairo_set_source_surface(cr, n->car.s, n->car.x, n->car.y);
6         cairo_paint(cr);
7     }
8 }
```

- draw_cars
  - it draws cars on the cairo window

Lab 4 - rbqueue.h

```
1  const int   BUFSIZE = 256;
2
3  class   RBQUEUE
4  {
5      private:
6          Person   buf[BUFSIZE];
7          int      front;
8          int      rear;
9          int      nextIndex(int index);
10
11     public:
```

- class RBQUEUE:
  - ring buffer queue for people

Lab 4 - rbqueue.h

```
1         RBQUEUE();
2         ~RBQUEUE();
3
4         bool     Insert(Person s);
5         bool     Remove(Person &s);
6         bool     isEmpty();
7         bool     isFull();
8         void     draw_people(cairo_t *cr);
9 };
```

Lab 4 - rbqueue.cpp

```
1  RBQUEUE::RBQUEUE(void)
2  {
3  }
4
5  RBQUEUE::~RBQUEUE(void)
6  {
7      while (front != rear)
8      {
9          cairo_surface_destroy(buf[front].s);
10         front = nextIndex(front);
11     }
12 }
```

- member functions of class RBQUEUE
- constructor and destructor
  - destructor deletes all the cairo surfaces

Lab 4 - rbqueue.cpp

```
1 int      RBQUEUE::nextIndex(int index)
2 {
3     if (++index == BUFSIZE)
4         index = 0;
5     return index;
6 }
```

- nextIndex
  - it returns the next index of the ring buffer queue

Lab 4 - rbqueue.cpp

```
1 bool     RBQUEUE::Insert(Person s)
2 {
3     if (isFull())
4         return false;
5     buf[rear] = s;
6     rear = nextIndex(rear);
7     return true;
8 }
```

- Insert
  - it takes a person struct and puts into the next buf
  - if the buf is full, then returns false

Lab 4 - rbqueue.cpp

```
1 bool     RBQUEUE::Remove(Person &s)
2 {
3     if (isEmpty())
4         return false;
5     s = buf[front];
6     front = nextIndex(front);
7     return true;
8 }
```

- Remove
  - it removes the front item and returns it

Lab 4 - rbqueue.cpp

```
1 bool     RBQUEUE::isEmpty()
2 {
3     return (front == rear);
4 }
```

- isEmpty
  - it checks if the buf is empty or not

Lab 4 - rbqueue.cpp

```
1 bool     RBQUEUE::isFull()
2 {
3     return (nextIndex(rear) == front);
4 }
```

- isFull
  - it checks if the buf if full or not

Lab 4 - rbqueue.cpp

```
1 void     RBQUEUE::draw_people(cairo_t* cr)
2 {
3     for(int i = front; i != rear; i = nextIndex(i))
4     {
5         cairo_set_source_surface(cr, buf[i].s, buf[i].x, buf[i].y);
6         cairo_paint(cr);
7     }
8 }
```

- draw_people
  - it draws the people on the cairo window

Lab 4 - main.cpp

```
1  LLQUEUE          cars;
2  RBQUEUE          people;
3  DRIVE            drive;
4  Fl_Cairo_Window  cw(WIDTH, HEIGHT);
5
6  int main(void)
7  {
8      srand(clock());
9      cw.label("Car Queues");
10     cw.set_draw_cb(draw_cb);
11     load_cars();
12     Fl::add_timeout(2, arrive_cb);
13     Fl::add_timeout(0.1, redraw_cb);
14     cw.show();
15     Fl::run();
16     cw.clear();
17     return (0);
18 }
```

- main.cpp
  - srand resets the rand() function
  - it creates the cairo window, loads cars, and adds drawing callback functions
  - arrive_cb will be called every 2 seconds
  - draw_cb will be called every 0.1 second

Lab 4 - load_cars.cpp

```
1  void load_cars(void)
2  {
3      for (int i = 0; i < MAX_CAR; i++)
4      {
5          Car      c;
6          string   s;
7          int      r;
8
9          r = rand() % 5;
10         s = CAR_PATH + to_string(r) + PNG;
11         c.s = cairo_image_surface_create_from_png(s.c_str());
12         c.x = 200 - (i * 50);
13         c.y = static_cast<double>(HEIGHT) * 0.85 + i * 5;
14         cars.Insert(c);
15     }
16 }
```

- load_cars.cpp
  - loads car images to car structs and inserts into the linked list queue
  - it sets the x,y position of cars

Lab 4 - draw_cb.cpp

```
1  void    draw_cb(Fl_Cairo_Window *cw, cairo_t *cr)
2  {
3      (void)cw;
4      if (cars.isEmpty() && !drive.move)
5          Fl::remove_timeout(redraw_cb);
6      draw_background(cr);
7      cars.draw_cars(cr);
8      people.draw_people(cr);
9      if (drive.move)
10         draw_drive(cr);
11     else
12         load_drive();
13 }
```

- draw_cb.cpp
  - it draws background, cars, and people
  - if drive.move is true, it calls draw_drive function
  - if all cars are gone, it removes this callback to stop drawing

Lab 4 - draw_background.cpp

```
1  void draw_background(cairo_t *cr)
2  {
3      double          w, h;
4      cairo_surface_t *s;
5
6      cairo_save(cr);
7      s = cairo_image_surface_create_from_png("streets/street2.png");
8      w = static_cast<double>(WIDTH) / cairo_image_surface_get_width(s);
9      h = static_cast<double>(HEIGHT) / cairo_image_surface_get_height(s);
10     cairo_scale(cr, w, h);
11     cairo_set_source_surface(cr, s, 0, 0);
12     cairo_paint(cr);
13     cairo_surface_destroy(s);
14     cairo_restore(cr);
15 }
```

- draw_background.cpp
  - it draws background
  - cairo_surface_destroy deletes loaded image after drawing

Lab 4 - load_drive.cpp

```
1  void     load_drive(void)
2  {
3      if (cars.isEmpty() || people.isEmpty())
4          return ;
5      cars.Remove(drive.car);
6      people.Remove(drive.person);
7      drive.st_x = drive.x = drive.car.x;
8      drive.st_y = drive.y = drive.car.y;
9      drive.move = true;
10 }
```

- load_drive.cpp
  - it takes a car and a person and sets the start position
  - if there's no car or no person, then it does nothing

Lab 4 - draw_drive.cpp

```
1  void     draw_drive(cairo_t *cr)
2  {
3      if (drive.x < WIDTH)
4      {
5          cairo_save(cr);
6          cairo_set_source_surface(cr, drive.car.s, drive.x, drive.y);
7          cairo_paint(cr);
8          cairo_scale(cr, 0.5, 0.5);
9          cairo_set_source_surface(cr, drive.person.s,
10                                   drive.x * 2 + 20, drive.y * 2 - 20);
11         cairo_paint(cr);
12         cairo_restore(cr);
13         drive.x += (DEST_X - drive.st_x) * 0.02;
14         drive.y += (DEST_Y - drive.st_y) * 0.02;
15     }
```

Lab 4 - draw_drive.cpp

```
1       else
2       {
3           cairo_surface_destroy(drive.car.s);
4           cairo_surface_destroy(drive.person.s);
5           drive.move = false;
6       }
7   }
```

- draw_drive.cpp
  - it draws the driving animation
  - it draws the car and the person and updates x,y position
  - 2% of the total distance is added every time
    - (from the start to the destination)
  - if it arrives the destination, then deletes it

Lab 4 - arrive_cb.cpp

```
1  void    arrive_cb(void *)
2  {
3      static int  n = 1;
4      Person      p;
5      string      s;
6      int         r;
7
8      r = rand() % 8 + 1;
9      s = PERSON_PATH + to_string(r) + PNG;
10     p.s = cairo_image_surface_create_from_png(s.c_str());
11     r = rand() % 10;
12     p.x = (WIDTH * 0.5) + r * (WIDTH * 0.04);
13     p.y = (HEIGHT * 0.9) - (r * 2);
14     people.Insert(p);
```

Lab 4 - arrive_cb.cpp

```
1      if (n++ == MAX_CAR)
2          Fl::remove_timeout(arrive_cb);
3      else
4          Fl::repeat_timeout(static_cast<double>(r / 3 + 2), arrive_cb);
5  }
```

- arrive_cb.cpp
  - it loads people
  - it places people on random place using rand()
  - resets callback time using rand()
  - it creates people the number of MAX_CAR

Lab 4 - redraw_cb.cpp

```
1 void     redraw_cb(void *)
2 {
3     cw.redraw();
4     Fl::repeat_timeout(0.1, redraw_cb);
5 }
```

- redraw_cb.cpp
  - this callback calls redraw every 0.1 second
  - redraw function calls the main drawing function, which is draw_cb
    - set by set_draw_cb function in the main