CS2263 Assignment 5 Report
Stephen Cole
3553803

The modules I used were:

BusRoute.h

A BusRoute contains a routeName that identifies it, a PointList that contains all the stops in that route and the number of stops in that route so that the stops can easily be looped through.

Student.h

This module is just a wrapper for the data taken in about each student, it only has one method that is createStudent and is mostly used just to improve readability. A Student has a location and a name.

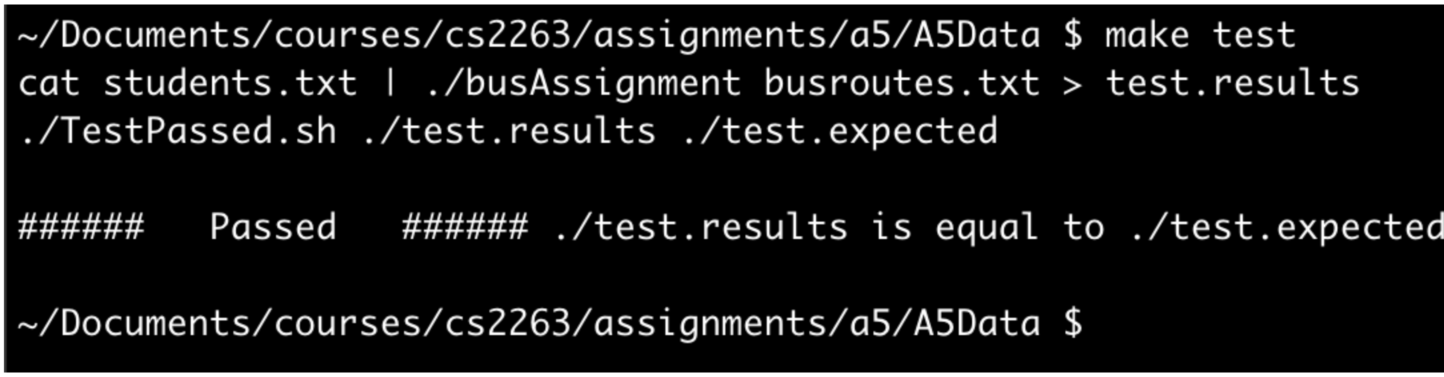
PointList.h

A PointList is a list of Point2D's, it's methods are just for memory allocation and accessing elements in the list. It is made up of a Point2D array and it's length.

Point2D.h

A Point2D is a 2D point containing an x and y value. This module contains getters and setters along with a distance between two points function that is used to find the closest bus route.

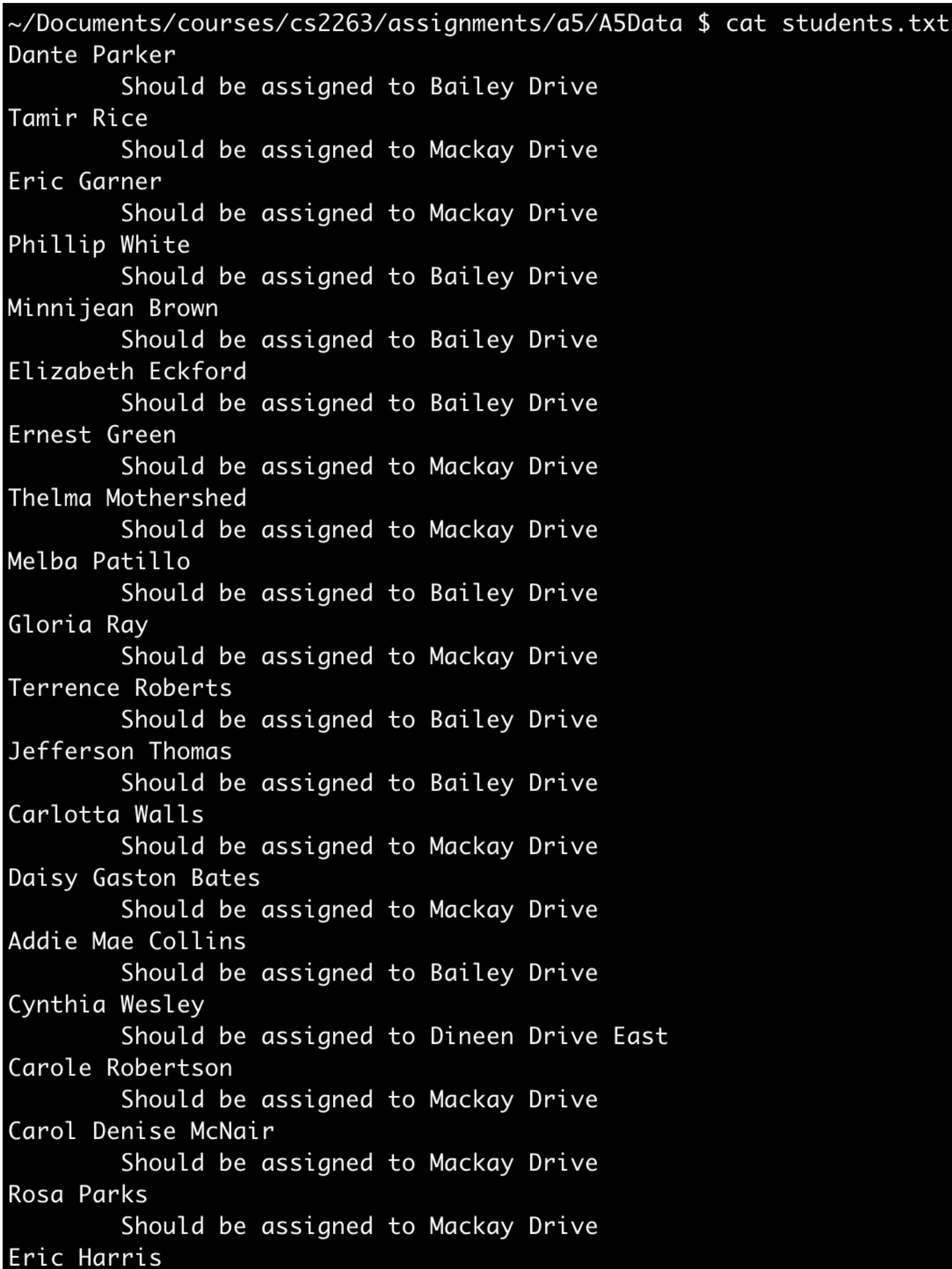
Test


```
~/Documents/courses/cs2263/assignments/a5/A5Data $ make test
cat students.txt | ./busAssignment busroutes.txt > test.results
./TestPassed.sh ./test.results ./test.expected

######   Passed   ###### ./test.results is equal to ./test.expected

~/Documents/courses/cs2263/assignments/a5/A5Data $
```

Output



```
~/Documents/courses/cs2263/assignments/a5/A5Data $ cat students.txt
Dante Parker
        Should be assigned to Bailey Drive
Tamir Rice
        Should be assigned to Mackay Drive
Eric Garner
        Should be assigned to Mackay Drive
Phillip White
        Should be assigned to Bailey Drive
Minnijean Brown
        Should be assigned to Bailey Drive
Elizabeth Eckford
        Should be assigned to Bailey Drive
Ernest Green
        Should be assigned to Mackay Drive
Thelma Mothershed
        Should be assigned to Mackay Drive
Melba Patillo
        Should be assigned to Bailey Drive
Gloria Ray
        Should be assigned to Mackay Drive
Terrence Roberts
        Should be assigned to Bailey Drive
Jefferson Thomas
        Should be assigned to Bailey Drive
Carlotta Walls
        Should be assigned to Mackay Drive
Daisy Gaston Bates
        Should be assigned to Mackay Drive
Addie Mae Collins
        Should be assigned to Bailey Drive
Cynthia Wesley
        Should be assigned to Dineen Drive East
Carole Robertson
        Should be assigned to Mackay Drive
Carol Denise McNair
        Should be assigned to Mackay Drive
Rosa Parks
        Should be assigned to Mackay Drive
Eric Harris
```

Source Code:

Point2D.h

```
/*
 *      Point2D.h file -- header file for two dimensional point data
 *
 * Original: Rick Wightman, June, 2020
 */


#ifndef POINT2D_H
#define POINT2D_H
#include <stdio.h>
#include <math.h>
typedef struct point2d
{
        double x;
        double y;

} Point2D;

/*
 * mallocPoint2D: allocates memory for a Point2D
 *
 * returns: pointer to allocated memory; NULL on fail
 */
Point2D* mallocPoint2D();

/*
 * freePoint2D: deallocates memory for a Point2D
 *
 * Parameters: Point2D* pPtThis - pointer to free
 *
 * returns: nothing
 */
void freePoint2D(Point2D* pPtThis);

Point2D* createPoint2D(double x, double y);

void setPoint2D(Point2D* pPtThis, double x, double y);

void setXPoint2D(Point2D* pPtThis, double x);
```

```
double getYPoint2D(Point2D* pPtThis);

Point2D* fscanfPoint2D(FILE* pFin);

double getDistancePoint2D( Point2D* ptThis, Point2D* pPtThat);

#endif
```

Point2D.c

```
/*
 *        Point2D.c file -- source file for two dimensional point data
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Point2D.h"

/*
 * mallocPoint2D: allocates memory for a Point2D
 *
 * returns: pointer to allocated memory; NULL on fail
 */
Point2D* mallocPoint2D(){
          Point2D* pPtThis = (Point2D*) malloc(sizeof(Point2D) );
          return pPtThis;
}

/*
 * freePoint2D: deallocates memory for a Point2D
 *
 * Parameters: Point2D* pPtThis - pointer to free
 *
 * returns: nothing
 */
void freePoint2D(Point2D* pPtThis){
          free(pPtThis);
}

Point2D* createPoint2D(double x, double y){
          Point2D* pPtThis = mallocPoint2D();
          if(pPtThis != (Point2D*)NULL ){
                    setPoint2D(pPtThis,x,y);
          }
          return pPtThis;
```

```c
}

void setPoint2D(Point2D* pPtThis, double x, double y){
        pPtThis->x = x;
        pPtThis->y = y;
}

void setXPoint2D(Point2D* pPtThis, double x){
        pPtThis->x = x;
}

double getYPoint2D(Point2D* pPtThis){
        return pPtThis->y;
}

Point2D* fscanfPoint2D(FILE* pFIn){
        Point2D* pPtThis;
        double x;
        double y;
        int iNRead;
        iNRead = fscanf(pFIn, "%lf %lf", &x, &y);
        if(iNRead !=2 ) return (Point2D*)NULL;
        pPtThis = createPoint2D(x, y);
        return pPtThis;
}

double getDistancePoint2D( Point2D* pPtThis, Point2D* pPtThat){
        double dX;
        double dY;
        double distance;
        dX = pPtThis->x - pPtThat->x;
        dY = pPtThis->y - pPtThat->y;
        distance = sqrt(dX*dX + dY*dY);
        return distance;
}
```

Student.h

```c
#ifndef STUDENT_H
#define STUDENT_H

#include <stdio.h>
#include "Point2D.h"

typedef struct student
{
	char* name;
	Point2D location;
} Student;

Student* createStudent(char* name, Point2D location);

#endif
```

Student.c

```
#include "Point2D.h"
#include "Student.h"

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


Student* createStudent(char* name, Point2D location)
{
	Student* stud = (Student*)malloc(sizeof(Student));
	stud->name = name;
	stud->location = location;

	return stud;
}
```

PointList.h

```
#include "Point2D.h"
#include <stdlib.h>

#ifndef POINTLIST_H
#define POINTLIST_H

typedef struct pointlist{
	int length;
	Point2D* pointList;
} PointList;

PointList* mallocPointList(int iNElements);

void freePointList(PointList* pList);

int setElementPointList(PointList* pList, Point2D point, int index);

Point2D* getElementPointList(PointList* pList, int index);
```

```
#endif

PointList.c

#include "Point2D.h"
#include <stdlib.h>

#ifndef POINTLIST_H
#define POINTLIST_H

typedef struct pointlist{
	int length;
	Point2D* pointList;
} PointList;

PointList* mallocPointList(int iNElements);

void freePointList(PointList* pList);

int setElementPointList(PointList* pList, Point2D point, int index);

Point2D* getElementPointList(PointList* pList, int index);

#endif
~/Documents/courses/cs2263/assignments/a5/A5Data $ cat PointList.c
#include "Point2D.h"
#include "PointList.h"
#include <stdlib.h>
#include <stdio.h>

PointList* mallocPointList(int iNElements)
{
	PointList* pList = (PointList*)malloc(sizeof(PointList));
	pList->pointList = (Point2D*)malloc(iNElements * sizeof(Point2D));
	pList->length = iNElements;

	for (int i=0; i < pList->length; i++) {
		pList->pointList[i] = *mallocPoint2D();
	}

	return pList;
}

void freePointList(PointList* pList)
{
        for (int i=0; i < pList->length; i++) {
    free(&(pList->pointList[i]));
        }

        free(pList);
}

int setElementPointList(PointList* pList, Point2D point, int index)
{
        // How do we tell if the element previously held a String?
  //if (pList->pointList[index] != (Point2D)NULL) {
  //  free(pList->pointList[index]);
 // }
  Point2D* p = createPoint2D(point.x, point.y);
  pList->pointList[index] = *p;
  return index;
}

Point2D* getElementPointList(PointList* pList, int index)
{
        return &pList->pointList[index];
}
```

BusRoute.h

```
// BusRoute.h
#ifndef BUSROUTE_H
#define BUSROUTE_H

#include <stdio.h>
#include "Point2D.h"
#include "PointList.h"

typedef struct busroute
{
        char* routeName;
        PointList pList;
        int numStops;
} BusRoute;

BusRoute* mallocBusRoute(int numStops);

void addPoint(BusRoute* route, Point2D point, int n);

Point2D* getPoint(BusRoute* route, int n);
```

```
#endif
```

BusRoute.c

```
#include "Point2D.h"
#include "BusRoute.h"

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

BusRoute* mallocBusRoute(int numStops)
{
	BusRoute* route = (BusRoute*)malloc(sizeof(BusRoute));
	route->pList = *mallocPointList(numStops);
	route->numStops = numStops;
	return route;
}

void addPoint(BusRoute* route, Point2D point, int n)
{
	setElementPointList(&route->pList, point, n);
}


Point2D* getPoint(BusRoute* route, int n)
{
	Point2D* p = getElementPointList(&route->pList, n);
	return p;
}
```

readRoutes.c

```
#include "PointList.h"
#include "Point2D.h"
#include "BusRoute.h"
#include "Student.h"

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

#define MAX_STRING_LENGTH 255

/**
 * You know what imo this code doesn't look as bad as normal
 * @StephenCole19
 */
int main(int argc, char** argv)
{
        FILE* fp;
        int numRoutes = 0;

        fp = fopen(argv[1], "r");
        fscanf(fp, "%d", &numRoutes);

        BusRoute** routes =  (BusRoute**)malloc(numRoutes * sizeof(BusRoute*));

        for(int i=0; i<numRoutes; i++)
        {
                int numStops = 0;
                fscanf(fp,"%d", &numStops);
                BusRoute* route = mallocBusRoute(numStops);

                for(int j=0; j<numStops; j++)
                {
                        double x;
                        double y;
                        fscanf(fp,"%lf %lf", &x, &y);
                        Point2D* point = createPoint2D(x,y);
                        addPoint(route, *point, j);
                }

                char *line = (char*)malloc(MAX_STRING_LENGTH);
                fgets(line, MAX_STRING_LENGTH, fp);
                line++;
                route->routeName = line;

                routes[i] = route;
        }

        double x = 0;
        double y = 0;


        while(fscanf(stdin, "%lf %lf", &x, &y) != 0)
        {
                char *name = (char*)malloc(MAX_STRING_LENGTH);
        if(fgets(name, MAX_STRING_LENGTH, stdin) == NULL)
                        break;
                name++;

                Point2D* p = createPoint2D(x,y);
        Student* stud = createStudent(name, *p);
        int routeIndex = 0;
        double shortestDistance = 1000000000000; // If it's longer than that he can fly to
school;
        for(int i=0; i<numRoutes; i++)
        {
            int numStops = routes[i]->numStops;
            for(int j=0; j<numStops; j++)
            {
                Point2D* currentStop = getPoint(routes[i],j);
                double currentDistance = getDistancePoint2D(currentStop, &stud-
>location);

                if(currentDistance < shortestDistance)
                {
                    routeIndex = i;
                    shortestDistance = currentDistance;
                }
            }

        }

        printf("%s\tShould be assigned to %s", stud->name, routes[routeIndex]-
>routeName);
        free(stud);
    }

    for(int i=0; i<numRoutes; i++)
        free(routes[i]);

    free(routes);


    return 0;
}
```