CS 2263 - FR01A

Assignment 5

By Ngoc Phuong Anh Nguyen - 3712361

November 2021

Question:

Write a C program to assign students to bus routes by selecting a particular bus route (set of bus stops) from a set bus routes. A bus route is an array of Point2D types with a route name (String). Bus routes are kept in a file and will need to be read in and held as an array of bus routes. A student will be represented with their location (Point2D) and name (String). The program will read in a file (name specified as a command line argument) of bus route information and store it. The program will then process students, read from standard in (no prompts), determine which route and stop is closest, and report the student and the route name and stop number.

Your program should use a module for lines (array of points and string) and this module should have the ability to determine the route/stop for the given student location (point). If you're feeling ambitious, you could even build a LineList module that manages a set of lines.

(a) In a few sentences describe each module that you design for your program. Focus on what each of the data structures holds and how each of the functions acts on them.

The program has Strings.c – the module holds functions that are used for strings. Inside this module, we have mallocString which allocates the requested memory and returns a pointer to it with the given string size, freeString which deallocates the memory of the string, and duplicateString which creates a duplicate string of the given string and return it.

Apart from it, there is another module called Point2D.c – the module holds functions that are used for Point2Ds. Inside this module, we have mallocPoint2D allocates the requested memory and returns a pointer to it, createPoint2D which creates a point with a given x and y, freePoint2D which deallocates the memory of the point, getXPoint2D which returns the x value of the point, getYPoint2D which returns the y value of the point, setPoint2D which edits the x and y of a point, then returns the point, and finally getDistancePoint2D which calculates the distance between 2 given points, then returns it.

Also, we have busAssignment.c – the main module. Inside, we have 2 struct functions, one for students, and one for lines. We also have fscanfBusRoute, which reads the file in the format "number of stops, list of stops, name of the route" and returns it in Line* datatype. freeLineList is a function that deallocates the memory of the line array. There is a main function that determines the nearest route/stop for the given student location (point). This function opens the files from standard input and command. Next, it calls fscanfBusRoute to read data from busRouteFile.txt and store it into an array. Then, the data in studentsFile.txt is read, but it is not stored. Meanwhile, for each student's data (x,y, and student name), they are stored temporarily in variables (studentLocation, studentName), and these variables are changed to each student. Also, for each student, there is a while loop to each route, then inside it is another while loop to each stop of the route. Inside that loop, we use getDistancePoint2D to calculate the distance between the student to the stops and compare them to each other to check if it is the nearest stop. If yes, then we use setPoint2D to update the point into nearestLocation, and duplicateString to copy the label of the route containing nearestLocation into nearestRoute. Finally, we print out nearestRoute and the coordinates of nearestLocation.

(b) Show the testing of one of the functions using a test program.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Point2D.h"

Point2D* mallocPoint2D()
{
  Point2D* pPtThis;
  pPtThis = (Point2D*) malloc(sizeof(Point2D));
  return pPtThis;
}

Point2D* createPoint2D(float x, float y)
{
  Point2D* pPtThis;
  pPtThis = mallocPoint2D();
  if(pPtThis == NULL)
  {
    return pPtThis;
  }
  pPtThis->x = x;
  pPtThis->y = y;
  return pPtThis;
}

double getXPoint2D(Point2D* pThis)
{
  return pThis->x;
}

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

void freePoint2D(Point2D* pThis)
{
  free(pThis);
}

int main(int argc, char **argv[])
{
  Point2D* pointA;
  double xA,yA;

  printf("Input x and y of point A: ");
  int iErrA = scanf("%lf%lf",&xA,&yA);
  while(iErrA != 2)
  {
    printf("Input x and y of point A again: ");
    iErrA = scanf("%lf%lf",&xA,&yA);
  }
  pointA = createPoint2D(xA, yA);

  printf("Point A: (%.2lf,%.2lf)\n",getXPoint2D(pointA), getYPoint2D(pointA));

  freePoint2D(pointA);
  return EXIT_SUCCESS;
}
```

Figure 1: Source Code of createPoint2D() function in a test program

```
[anguyen5@gc112m30 Assignment 5]$ gcc -c test.c
[anguyen5@gc112m30 Assignment 5]$ gcc -o test test.c
[anguyen5@gc112m30 Assignment 5]$ ./test
Input x and y of point A: 2 3.5
Point A: (2.00,3.50)
[anguyen5@gc112m30 Assignment 5]$
```

Figure 2: Output of the test program.

(c) Show the output from running your program on the included test data.

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "Strings.h"

String mallocString(int stringsize)
{
  return (String)malloc(sizeof(char) * (stringsize + 1));
}

void freeString(String s)
{
  free(s);
}

String duplicateString(String s)
{
  String sCopy = mallocString(strlen(s));
  if(sCopy != (String)NULL)
  {
    strcpy(sCopy,s);
  }
  return sCopy;
}
```

Figure 3: Source Code of Strings.c

```c
#ifndef STRINGS_H
#define STRINGS_H
#include <string.h>
typedef char* String;

String mallocString(int stringsize);
void freeString(String s);
String duplicateString(String s);

#endif
```

Figure 4: Source Code of Strings.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Point2D.h"

Point2D* mallocPoint2D()
{
  Point2D* pPtThis;
  pPtThis = (Point2D*) malloc(sizeof(Point2D));
  return pPtThis;
}

Point2D* createPoint2D(float x, float y)
{
  Point2D* pPtThis;
  pPtThis = mallocPoint2D();
  if(pPtThis == NULL)
  {
    return pPtThis;
  }
  pPtThis->x = x;
  pPtThis->y = y;
  return pPtThis;
}

void freePoint2D(Point2D* pThis)
{
  free(pThis);
}

double getXPoint2D(Point2D* pThis)
{
  return pThis->x;
}

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

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

double getDistancePoint2D(Point2D* pThis, Point2D* pThat)
{
  return sqrt(pow((getXPoint2D(pThis) - getXPoint2D(pThat)), 2)
  + pow((getYPoint2D(pThis) - getYPoint2D(pThat)), 2));
}
```

Figure 5: Source Code of Point2D.c

```h
#ifndef POINT2D_H
#define POINT2D_H

typedef struct point2D
{
  double x;
  double y;
}
Point2D;

Point2D* mallocPoint2D();
Point2D* createPoint2D(float x, float y);
void freePoint2D(Point2D* pThis);
double getXPoint2D(Point2D* pThis);
double getYPoint2D(Point2D* pThis);
void setPoint2D(Point2D* pPt, double x, double y);
double getDistancePoint2D(Point2D* pThis, Point2D* pThat);

#endif
```

Figure 6: Source Code of Point2D.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "Strings.h"
#include "Point2D.h"

typedef struct student
{
  Point2D* point;
  String name;
}
Student;

typedef struct line
{
  int size;
  String label;
  Point2D* points[];
}
Line;

void freeLineList(Line* lineList[])
{
  int size = (sizeof(lineList) / sizeof(lineList[0]));
  int i,j;
  for(i = 0; i < size; i++)
  {
    freeString(lineList[i]->label);
    for(j = 0; j < lineList[i]->size; j++)
    {
      freePoint2D(lineList[i]->points[j]);
    }
    free(lineList[i]->points);
    free(lineList[i]);
  }
  free(lineList);
}

Line* fscanfBusRoute(FILE* file, Line* line)
{
  int size;
  fscanf(file,"%d", &size);
  String s = mallocString(50);
  line = malloc(sizeof(Line*) + size*sizeof(Point2D*));
  line->label = mallocString(50);
  line->size = size;
  int j = 0;
  double x,y;
  while(j < line->size)
  {
    fscanf(file,"%lf %lf ",&x,&y);
    line->points[j] = createPoint2D(x,y);
    j++;
  }
  fgets(s,50,file);
  line->label = duplicateString(s);
  freeString(s);
  return line;
}

int main(int argc, char const *argv[])
{
  FILE* studentsFile = stdin;
  FILE* busRouteFile = fopen(argv[1], "r");

  if(studentsFile == (FILE*)NULL)
  {
    fprintf(stderr, "Unable to open file %s\n", stdin);
    return EXIT_FAILURE;
```

Figure 7: Source Code of busAssignment.c (1,2)

```c
  }
  if(busRouteFile == (FILE*)NULL)
  {
    fprintf(stderr, "Unable to open file %s\n", argv[1]);
    return EXIT_FAILURE;
  }

  int i = 0,j;
  int numberOfRoutes;
  fscanf(busRouteFile,"%d", &numberOfRoutes);
  Line* lineList[numberOfRoutes];
  String studentName = mallocString(50);
  String nearestRoute = mallocString(60);

  while( i < numberOfRoutes)
  {
    lineList[i] = fscanfBusRoute(busRouteFile,lineList[i]);
    i++;
  }

  double x = 0,y=0;
  double nearestDistance;
  double distance = 0;
  Point2D* nearestLocation = createPoint2D(x,y);
  Point2D* studentLocation = createPoint2D(x,y);
  while(fscanf(studentsFile,"%lf %lf ",&x,&y)==2)
  {
    distance = 0;
    fgets(studentName,50,studentsFile);
    printf("---------------\n%s", studentName);//for each student
    setPoint2D(studentLocation,x,y);
    printf("Location: %.2lf %.2lf\n", getXPoint2D(studentLocation),
    getYPoint2D(studentLocation));
    i=0;
    setPoint2D(nearestLocation, getXPoint2D(studentLocation),
    getYPoint2D(studentLocation));
    nearestDistance = 1000000;
    while(i < numberOfRoutes)//for each route
    {
      j = 0;
      while(j < lineList[i]->size)//for each bus stop
      {
        distance = getDistancePoint2D(studentLocation, lineList[i]->points[j]);
        if(nearestDistance >= distance)
        {
          nearestDistance = distance;
          setPoint2D(nearestLocation, getXPoint2D(lineList[i]->points[j]),
          getYPoint2D(lineList[i]->points[j]));
          nearestRoute = duplicateString(lineList[i]->label);
        }
        j++;
      }
      i++;
    }
    printf("Nearest route: %s",nearestRoute);
    printf("Nearest stop: %.2lf %.2lf\n", getXPoint2D(nearestLocation),
    getYPoint2D(nearestLocation));
  }

  freeString(studentName);
  freeString(nearestRoute);
  freePoint2D(nearestLocation);
  freePoint2D(studentLocation);

  freeLineList(lineList);

  fclose(busRouteFile);
  fclose(studentsFile);
  return 0;
}
```

Figure 8: Source Code of busAssignment.c (3,4)

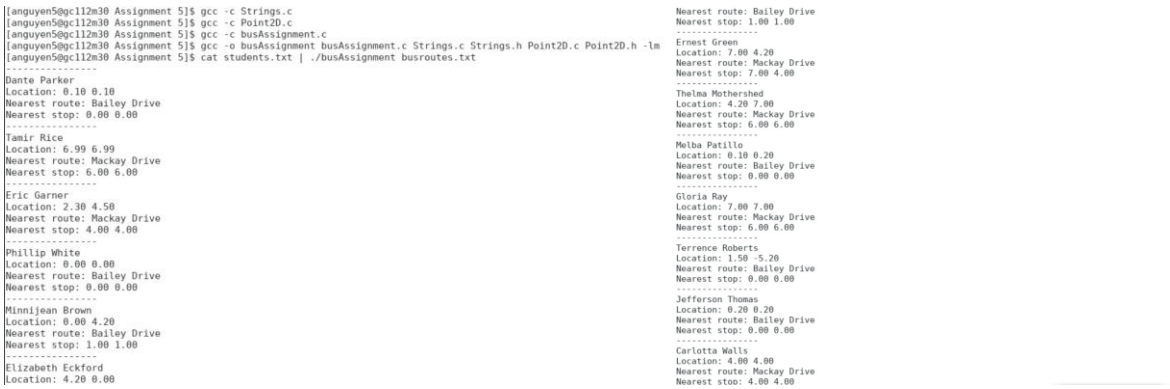

Figure 9: Output of busAssignment.c (1,2)

Figure 10: Output of busAssignment.c (3,4)

Note: There is an error after the output.

```
*** Error in `./busAssignment': double free or corruption (out): 0x0000000002334330 ***
======= Backtrace: =========
/lib64/libc.so.6(+0x81329)[0x7f756242c329]
./busAssignment[0x4009ca]
./busAssignment[0x4010b0]
/lib64/libc.so.6(__libc_start_main+0xf5)[0x7f75623cd555]
./busAssignment[0x400849]
======= Memory map: ========
00400000-00402000 r-xp 00000000 00:2a 10737570060                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 5/busAssignment
00601000-00602000 r--p 00001000 00:2a 10737570060                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 5/busAssignment
00602000-00603000 rw-p 00002000 00:2a 10737570060                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 5/busAssignment
02334000-02355000 rw-p 00000000 00:00 0                                  [heap]
7f755c000000-7f755c021000 rw-p 00000000 00:00 0
7f755c021000-7f7560000000 ---p 00000000 00:00 0
7f7562195000-7f75621aa000 r-xp 00000000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7f75621aa000-7f75623a9000 ---p 00015000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7f75623a9000-7f75623aa000 r--p 00014000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7f75623aa000-7f75623ab000 rw-p 00015000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7f75623ab000-7f756256f000 r-xp 00000000 fd:00 237296                     /usr/lib64/libc-2.17.so
7f756256f000-7f756276e000 ---p 001c4000 fd:00 237296                     /usr/lib64/libc-2.17.so
7f756276e000-7f7562772000 r--p 001c3000 fd:00 237296                     /usr/lib64/libc-2.17.so
7f7562772000-7f7562774000 rw-p 001c7000 fd:00 237296                     /usr/lib64/libc-2.17.so
7f7562774000-7f7562779000 rw-p 00000000 00:00 0
7f7562779000-7f756287a000 r-xp 00000000 fd:00 237316                     /usr/lib64/libm-2.17.so
7f756287a000-7f7562a79000 ---p 00101000 fd:00 237316                     /usr/lib64/libm-2.17.so
7f7562a79000-7f7562a7a000 r--p 00100000 fd:00 237316                     /usr/lib64/libm-2.17.so
7f7562a7a000-7f7562a7b000 rw-p 00101000 fd:00 237316                     /usr/lib64/libm-2.17.so
7f7562a7b000-7f7562a9d000 r-xp 00000000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7f7562c75000-7f7562c78000 rw-p 00000000 00:00 0
7f7562c96000-7f7562c9c000 rw-p 00000000 00:00 0
7f7562c9c000-7f7562c9d000 r--p 00021000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7f7562c9d000-7f7562c9e000 rw-p 00022000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7f7562c9e000-7f7562c9f000 rw-p 00000000 00:00 0
7ffcf4113000-7ffcf4134000 rw-p 00000000 00:00 0                          [stack]
7ffcf416b000-7ffcf416d000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
Abort
```