Assignment 5

Description

BRModule:

BRModule is the bus route module. It is a struct that has a routeName, a point list, and the number of stops on each route. It has a function that will malloc enough memory for the bus route. It also has a function that will add the point scanned to the bus route. Its final function is one that has the ability to get a point on the route using the index.

Point2D module:

It is a module that has an x and y coordinate for each point. The x and y are double values. Regarding the functions, it has one that will malloc enough memory for the 2d point. It also has one that can free the malloced memory. It has a function that can create a point which will come in handy in other functions. It has one that is able to change the x and y values of an already existing point. It has a function that can modify the x value of an already existing point. It has one that can return the y value of a point. It has a function that can scan a point directly from a file. It also has one that is able to calculate the distance between 2 points. Its final function can copy a point.

PointList Module

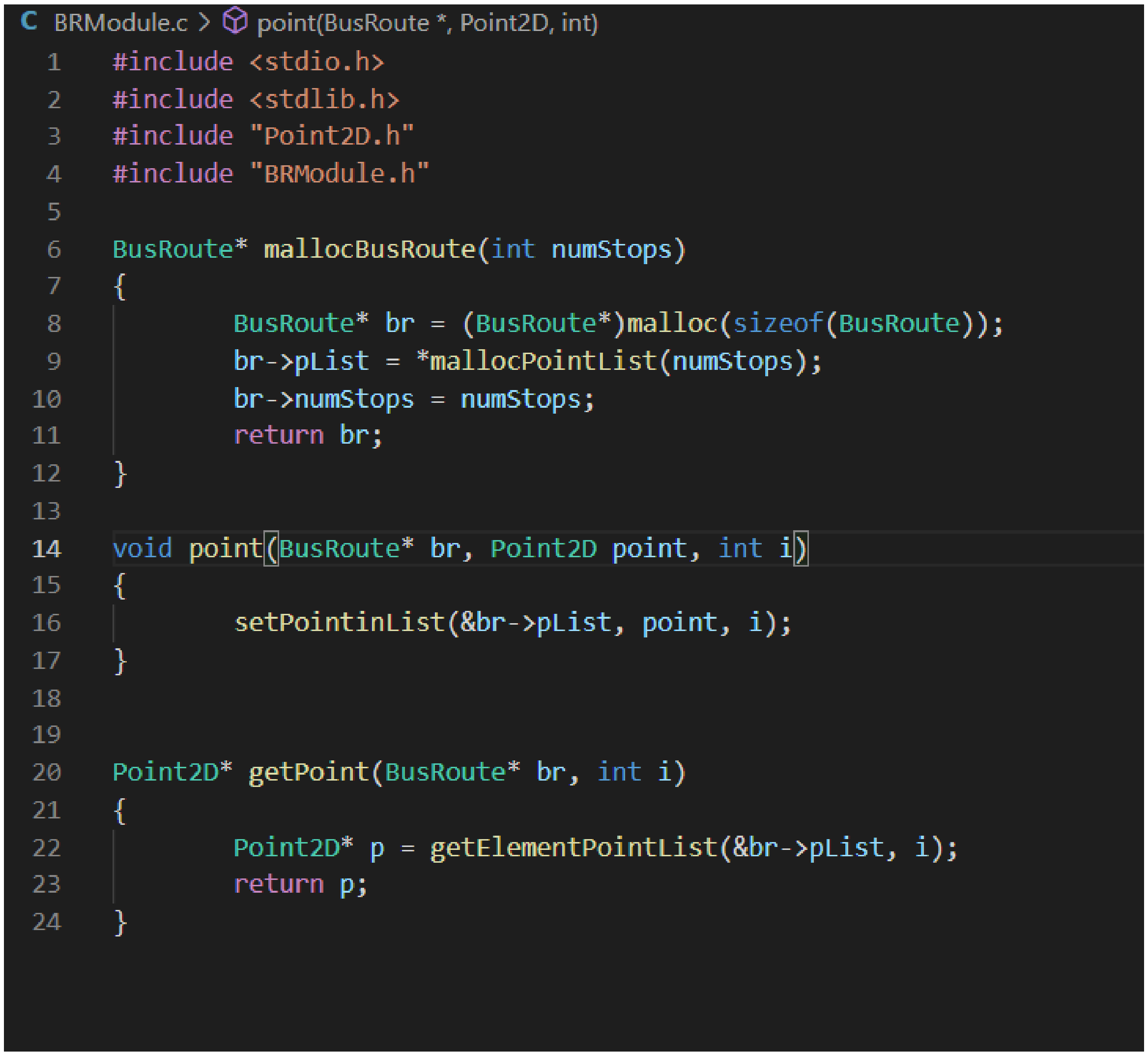
This module has a point list. Each point list has a length and a 2d point array. This module has a function that mallocs enough memory for the whole list. On the other hand, it has a function that frees the malloced memory. It has a function that can change/modify a point that already exists in the list. It also has a function that can return an element in the list.

Strings Module

The string is an easier, more comprehendible way to look at a char*. Similarly, the stringlist is an easier and more comprehendible way to look at the char**. It has a function that can malloc enough memory for the string. It a function that can free the malloced memory. It has a function that can duplicate a given string. It has a function that, similarly, can duplicate a string array, aka stringlist. It has a function that can compare 2 given strings. It has a function that scans a whole line in the command line (terminal) even with spaces.

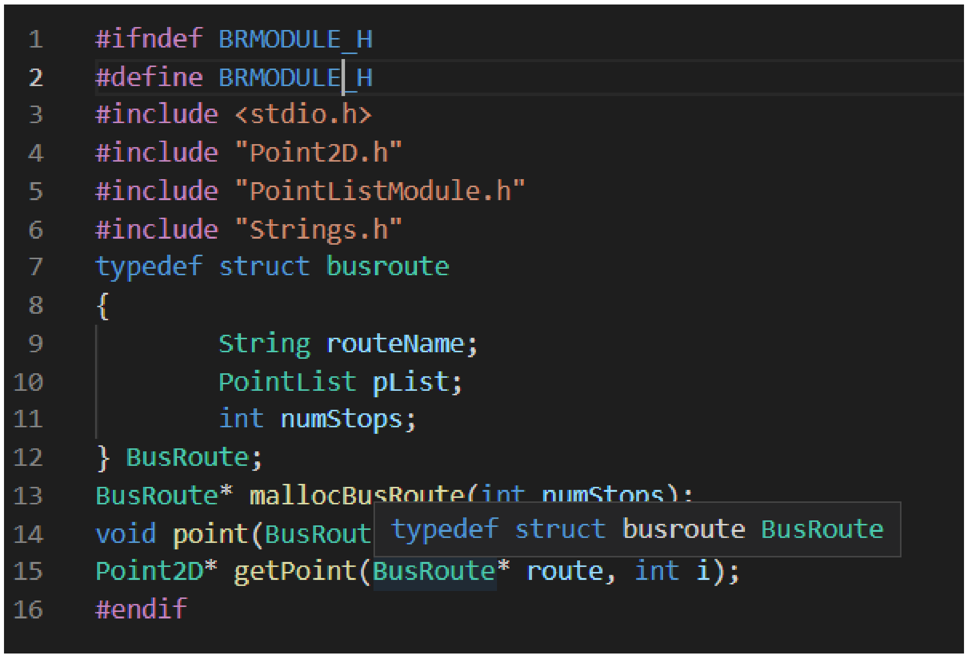
Source code

BrModule.c



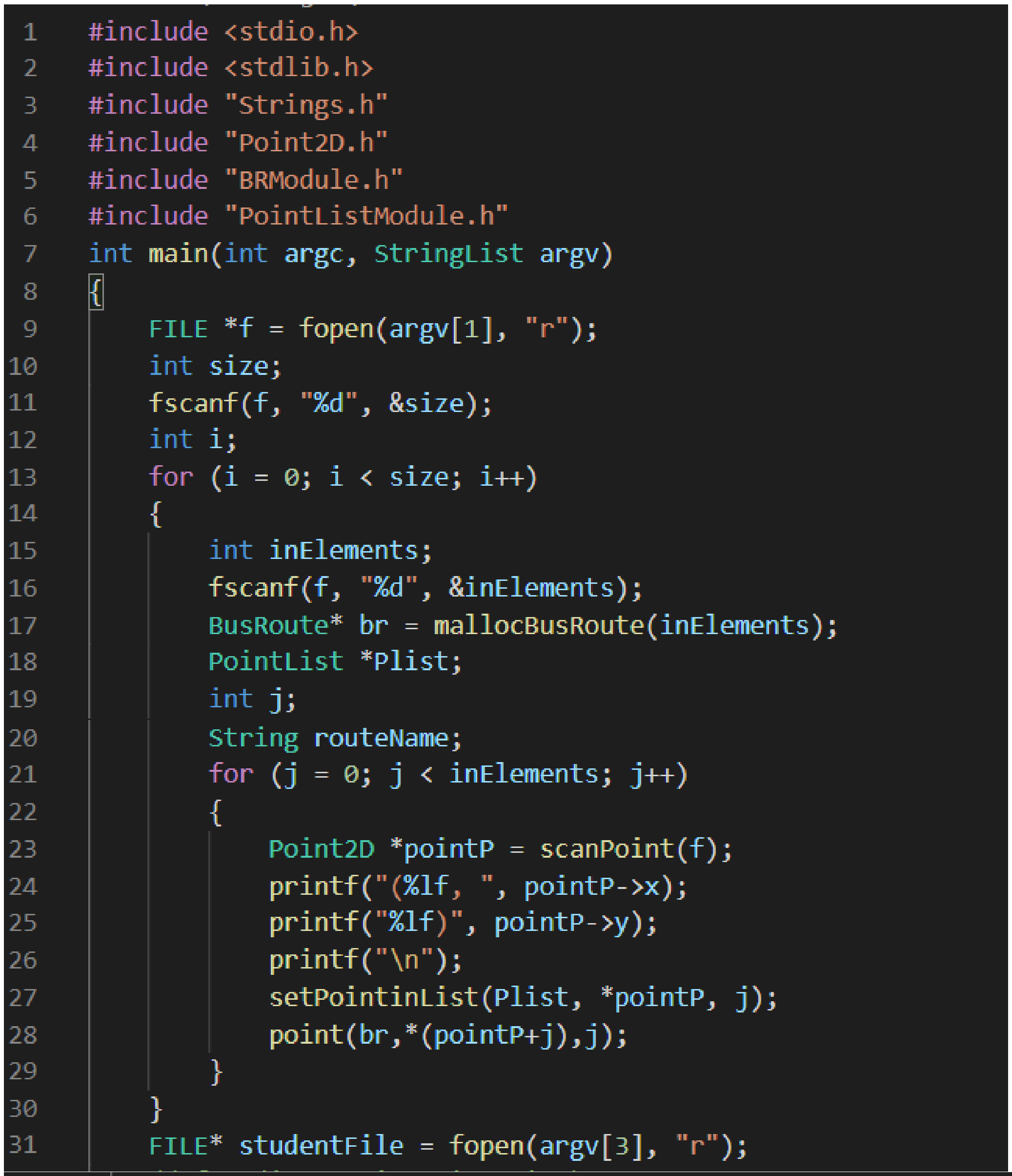
```c
#include <stdio.h>
#include <stdlib.h>
#include "Point2D.h"
#include "BRModule.h"

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

void point(BusRoute* br, Point2D point, int i)
{
        setPointinList(&br->pList, point, i);
}


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

BRModule.h

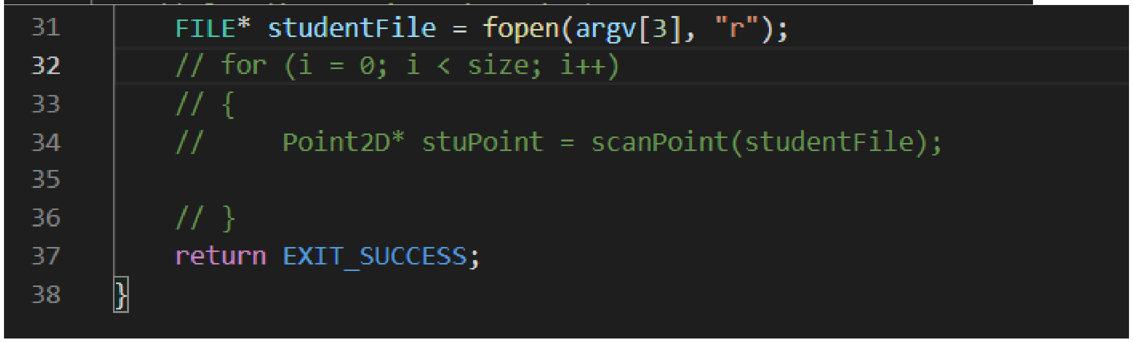


```c
#ifndef BRMODULE_H
#define BRMODULE_H
#include <stdio.h>
#include "Point2D.h"
#include "PointListModule.h"
#include "Strings.h"
typedef struct busroute
{
        String routeName;
        PointList pList;
        int numStops;
} BusRoute;
BusRoute* mallocBusRoute(int numStops);
void point(BusRout typedef struct busroute BusRoute
Point2D* getPoint(BusRoute* route, int i);
#endif
```

A5.c



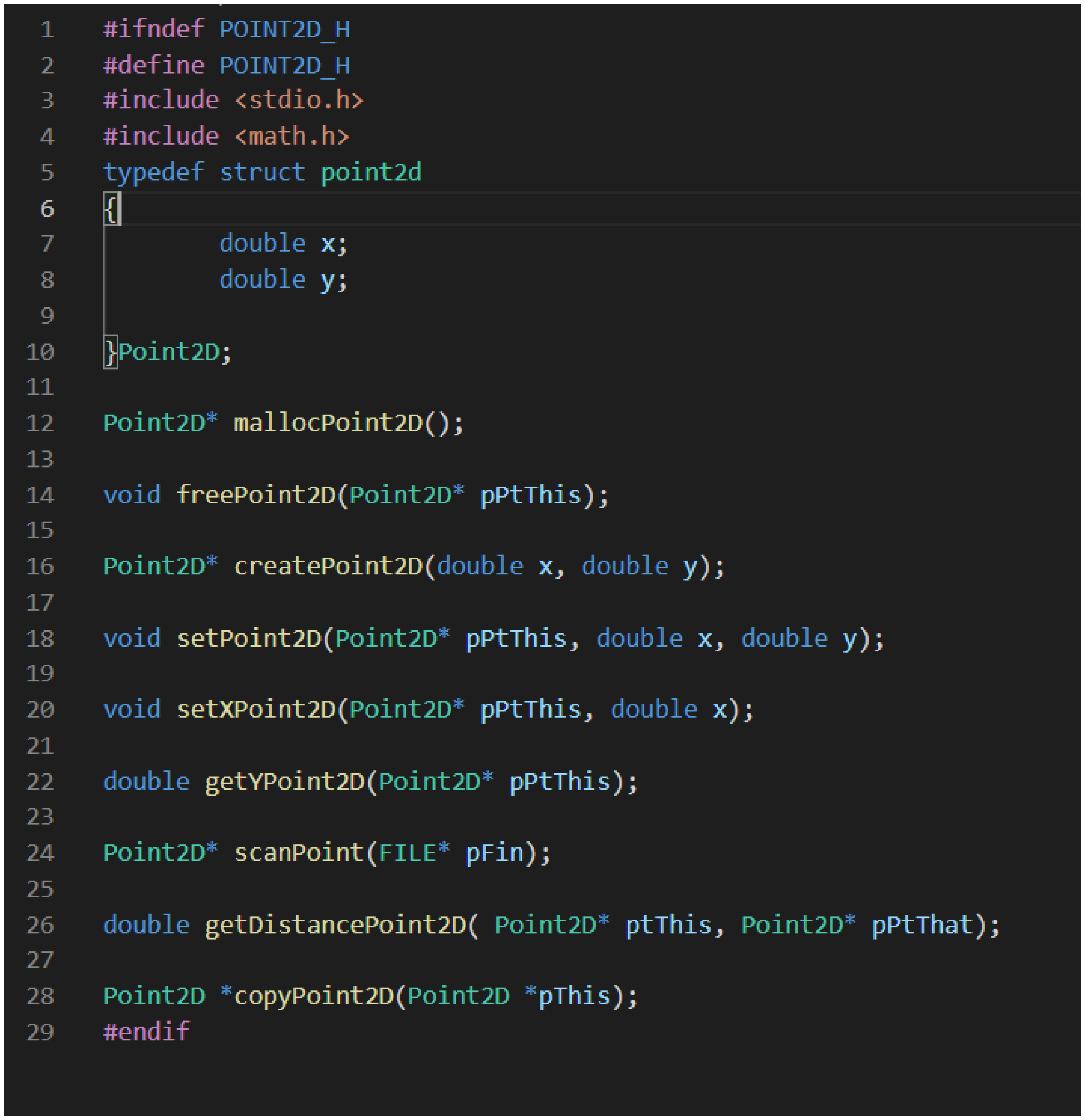
```c
#include <stdio.h>
#include <stdlib.h>
#include "Strings.h"
#include "Point2D.h"
#include "BRModule.h"
#include "PointListModule.h"
int main(int argc, StringList argv)
{
    FILE *f = fopen(argv[1], "r");
    int size;
    fscanf(f, "%d", &size);
    int i;
    for (i = 0; i < size; i++)
    {
        int inElements;
        fscanf(f, "%d", &inElements);
        BusRoute* br = mallocBusRoute(inElements);
        PointList *Plist;
        int j;
        String routeName;
        for (j = 0; j < inElements; j++)
        {
            Point2D *pointP = scanPoint(f);
            printf("(%lf, ", pointP->x);
            printf("%lf)", pointP->y);
            printf("\n");
            setPointinList(Plist, *pointP, j);
            point(br,*(pointP+j),j);
        }
    }
    FILE* studentFile = fopen(argv[3], "r");
```



```c
    FILE* studentFile = fopen(argv[3], "r");
```

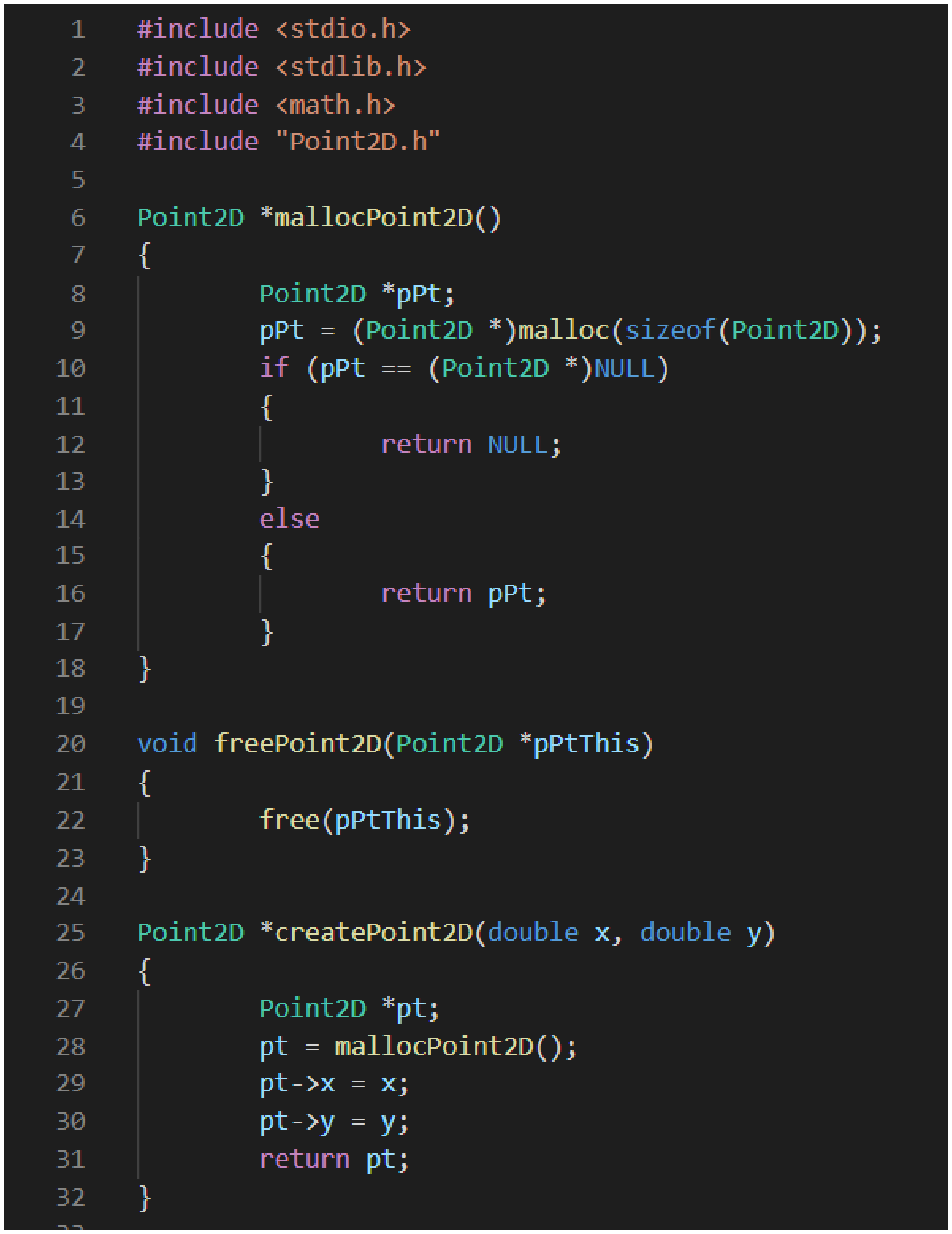
```c
    // for (i = 0; i < size; i++)
    // {
    //     Point2D* stuPoint = scanPoint(studentFile);

    // }
    return EXIT_SUCCESS;
}
```

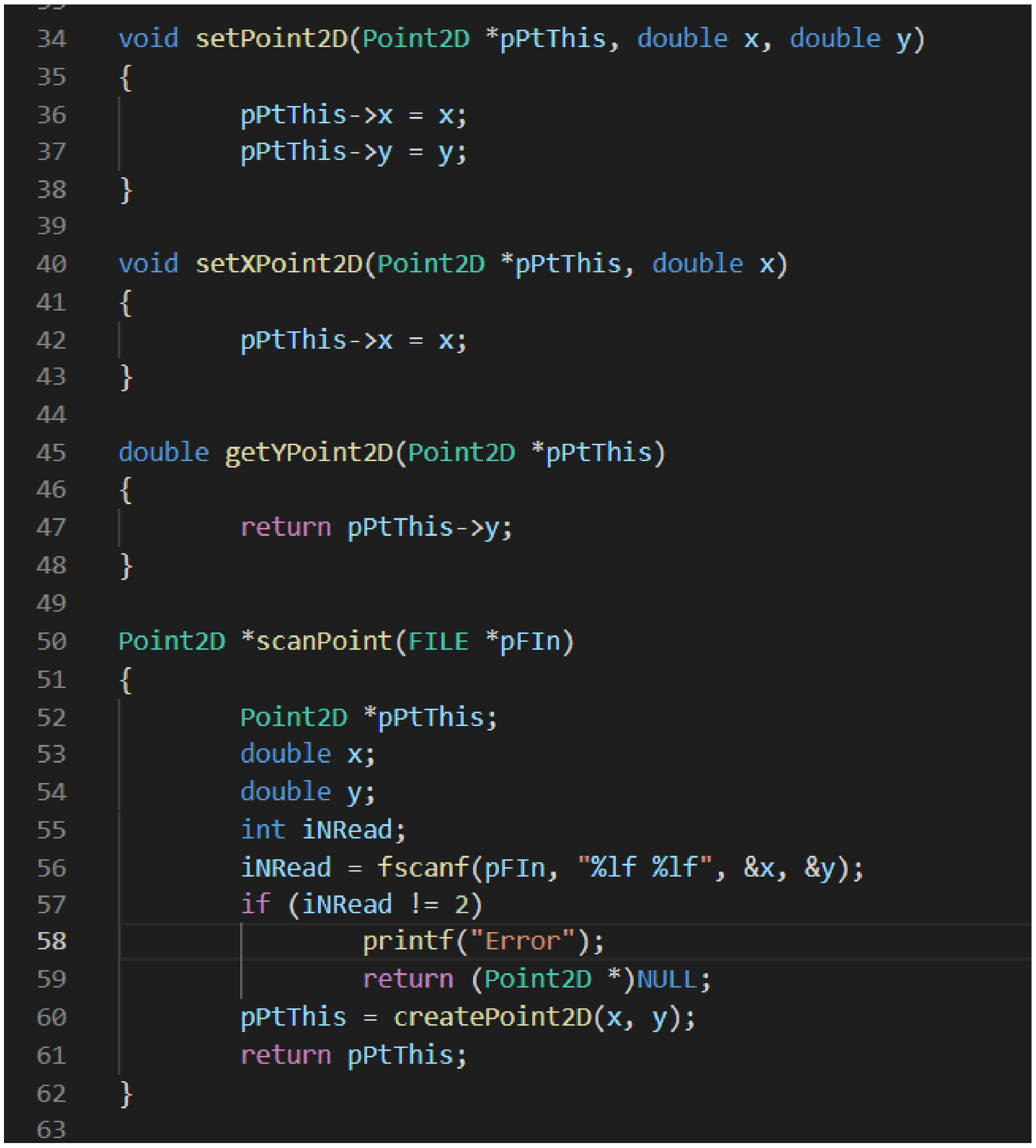
Point2D.h



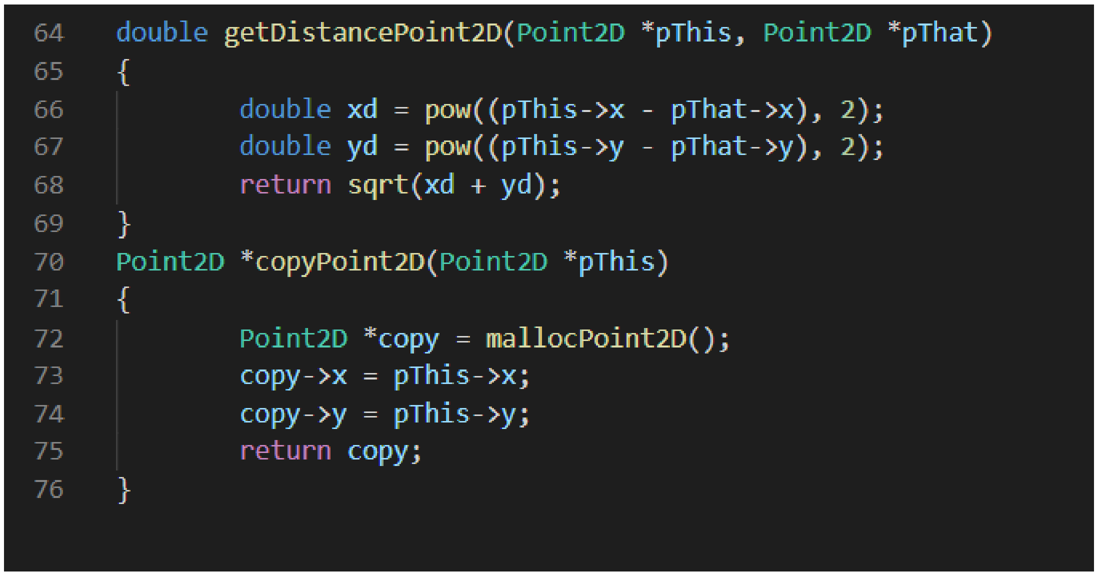
```c
#ifndef POINT2D_H
#define POINT2D_H
#include <stdio.h>
#include <math.h>
typedef struct point2d
{
        double x;
        double y;

}Point2D;

Point2D* mallocPoint2D();

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* scanPoint(FILE* pFin);

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

Point2D *copyPoint2D(Point2D *pThis);
#endif
```

Point2D.c


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

Point2D *mallocPoint2D()
{
        Point2D *pPt;
        pPt = (Point2D *)malloc(sizeof(Point2D));
        if (pPt == (Point2D *)NULL)
        {
                return NULL;
        }
        else
        {
                return pPt;
        }
}

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

Point2D *createPoint2D(double x, double y)
{
        Point2D *pt;
        pt = mallocPoint2D();
        pt->x = x;
        pt->y = y;
        return pt;
}
```

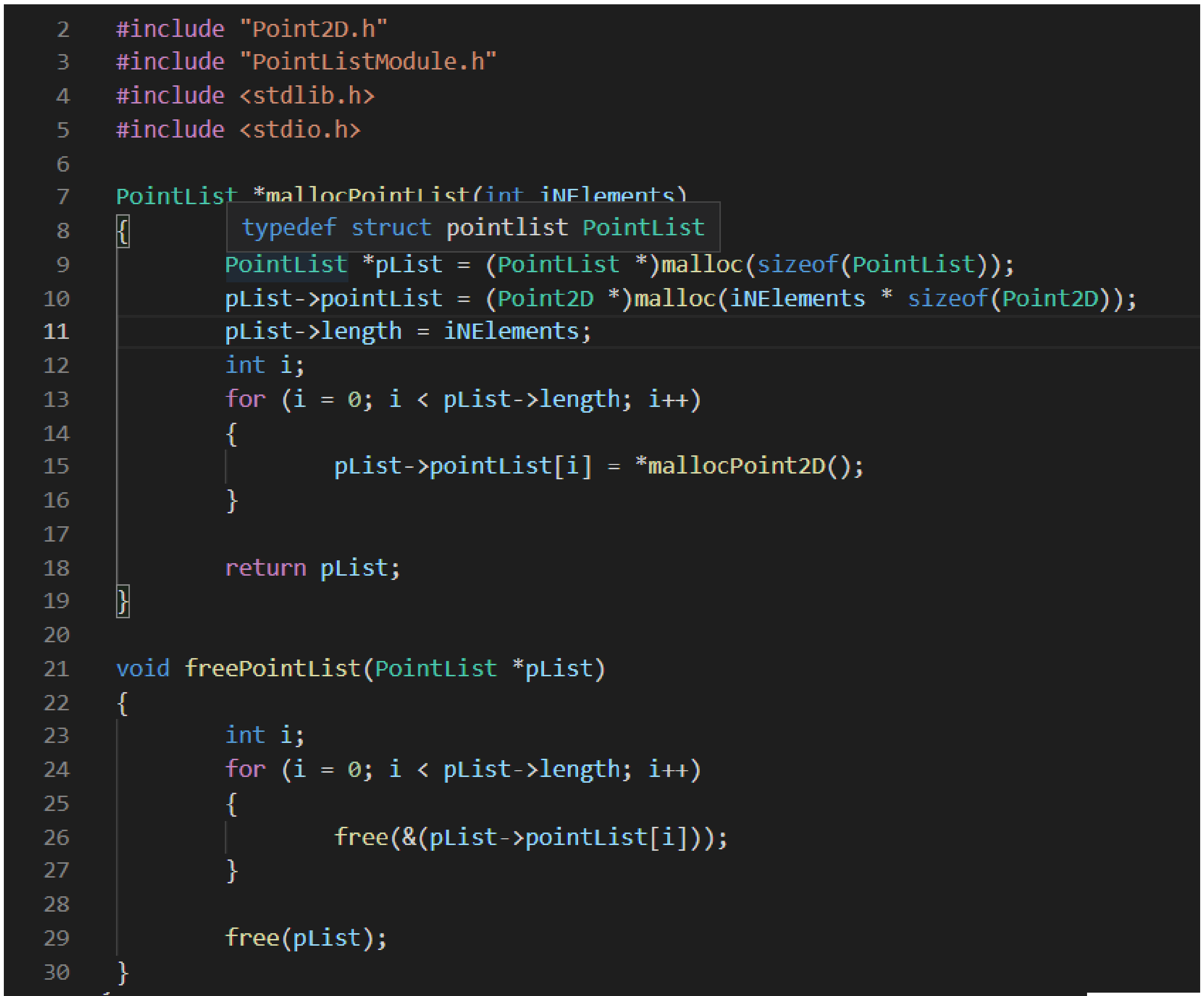


```c
void setPoint2D(Point2D *pPtThis, double x, double y)
{
        pPtThis->x = x;
        pPtThis->y = y;
}

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

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

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

```

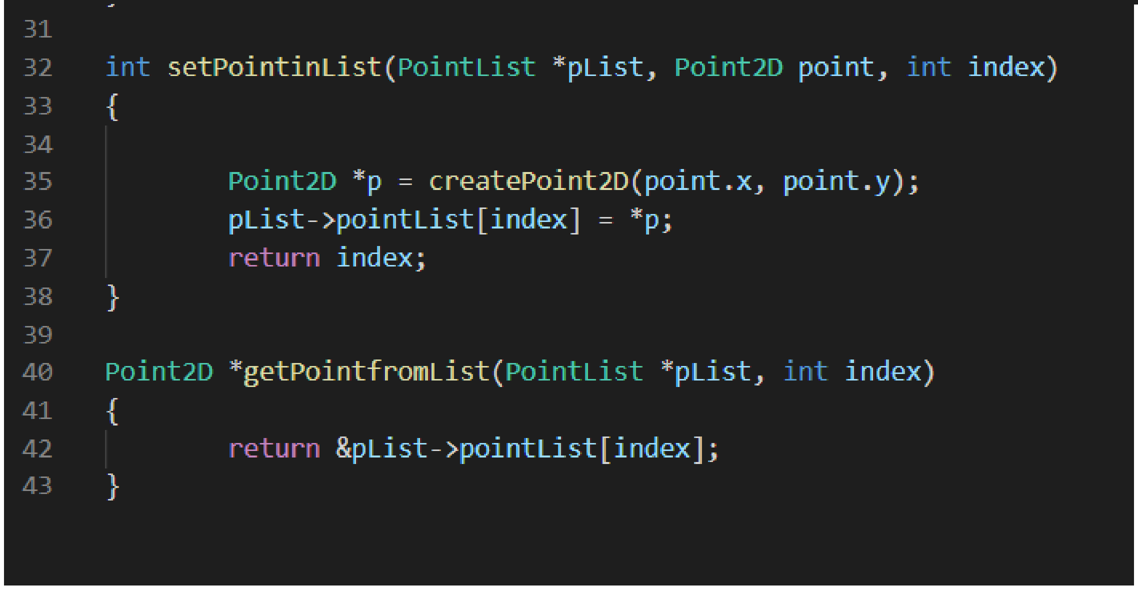


```
double getDistancePoint2D(Point2D *pThis, Point2D *pThat)
{
        double xd = pow((pThis->x - pThat->x), 2);
        double yd = pow((pThis->y - pThat->y), 2);
        return sqrt(xd + yd);
}
Point2D *copyPoint2D(Point2D *pThis)
{
        Point2D *copy = mallocPoint2D();
        copy->x = pThis->x;
        copy->y = pThis->y;
        return copy;
}
```

PointListModule.c

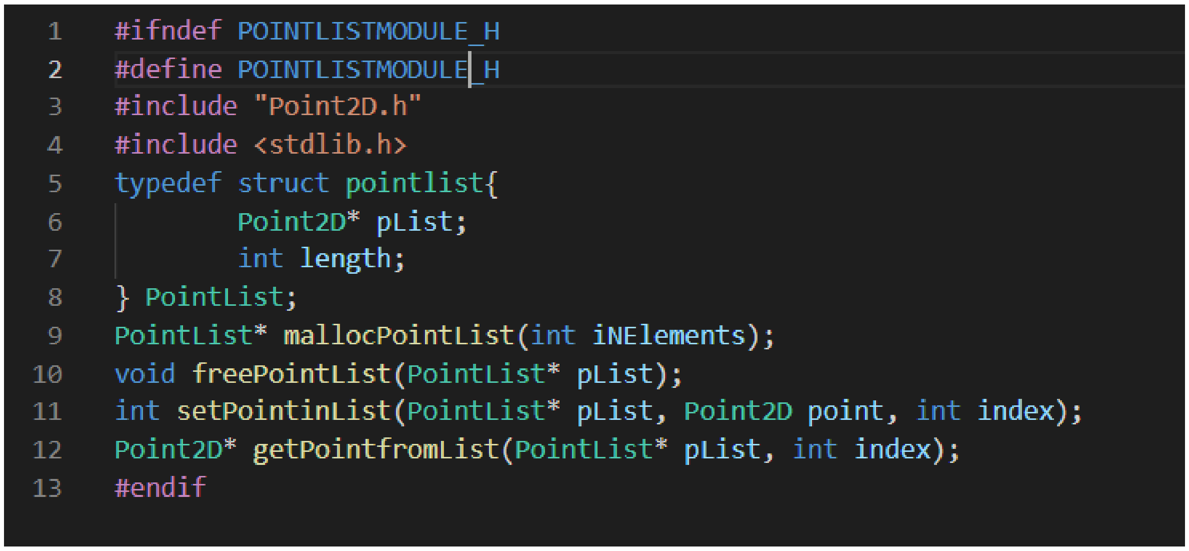


```c
#include "Point2D.h"
#include "PointListModule.h"
#include <stdlib.h>
#include <stdio.h>

PointList *mallocPointList(int iNElements)
{
        PointList *pList = (PointList *)malloc(sizeof(PointList));
        pList->pointList = (Point2D *)malloc(iNElements * sizeof(Point2D));
        pList->length = iNElements;
        int i;
        for (i = 0; i < pList->length; i++)
        {
                pList->pointList[i] = *mallocPoint2D();
        }

        return pList;
}

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

        free(pList);
}
```



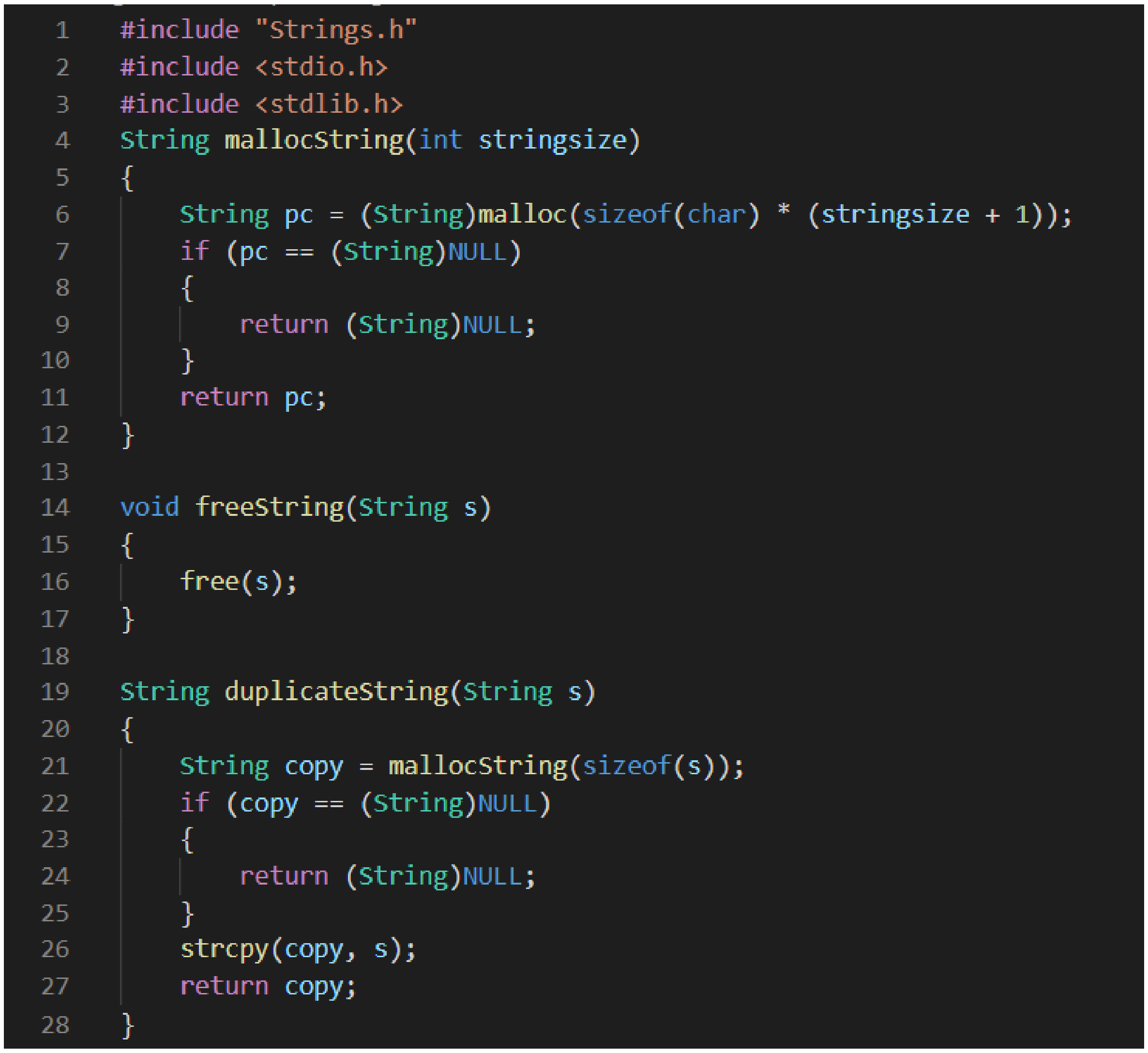
```c

int setPointinList(PointList *pList, Point2D point, int index)
{

        Point2D *p = createPoint2D(point.x, point.y);
        pList->pointList[index] = *p;
        return index;
}

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

PointListModule.h

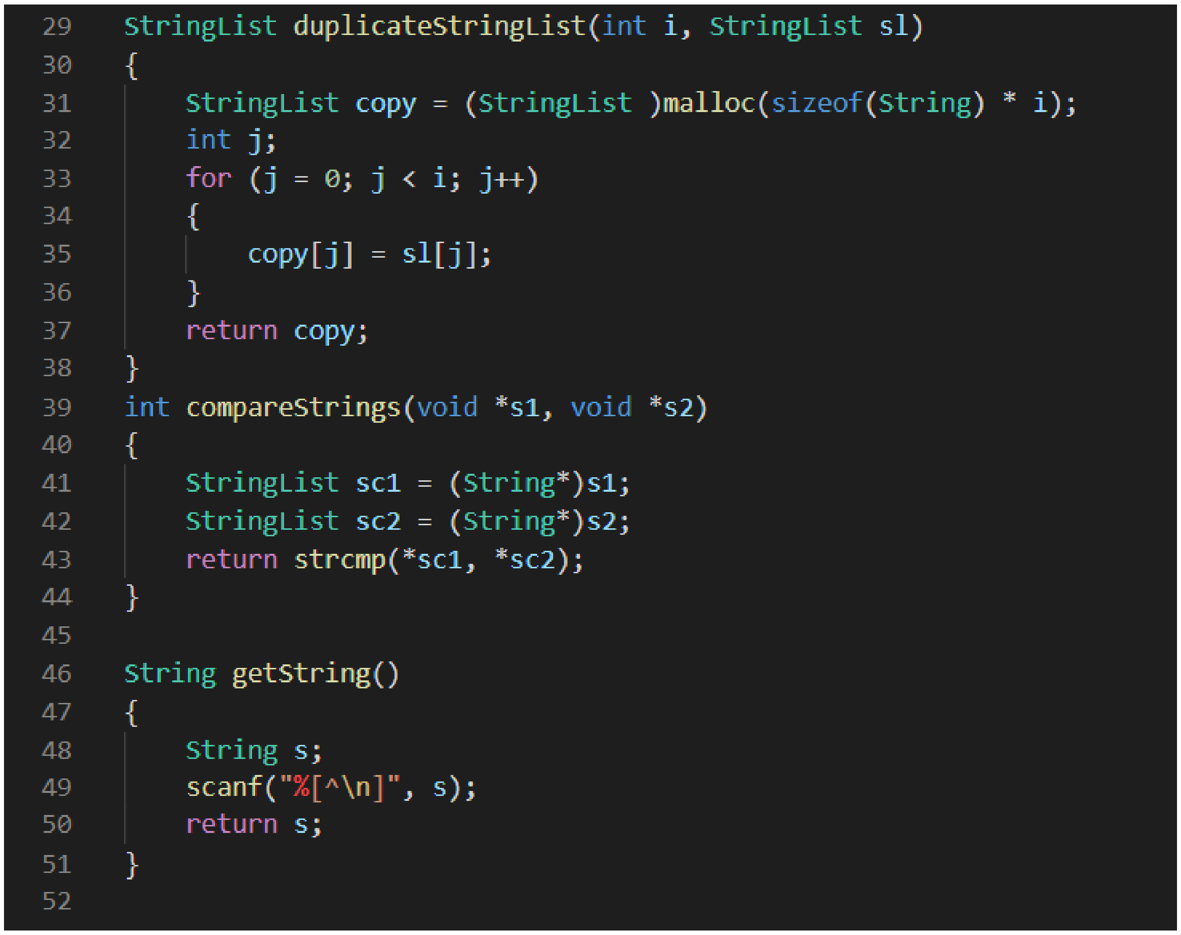


```c
#ifndef POINTLISTMODULE_H
#define POINTLISTMODULE_H
#include "Point2D.h"
#include <stdlib.h>
typedef struct pointlist{
        Point2D* pList;
        int length;
} PointList;
PointList* mallocPointList(int iNElements);
void freePointList(PointList* pList);
int setPointinList(PointList* pList, Point2D point, int index);
Point2D* getPointfromList(PointList* pList, int index);
#endif
```

Strings.c

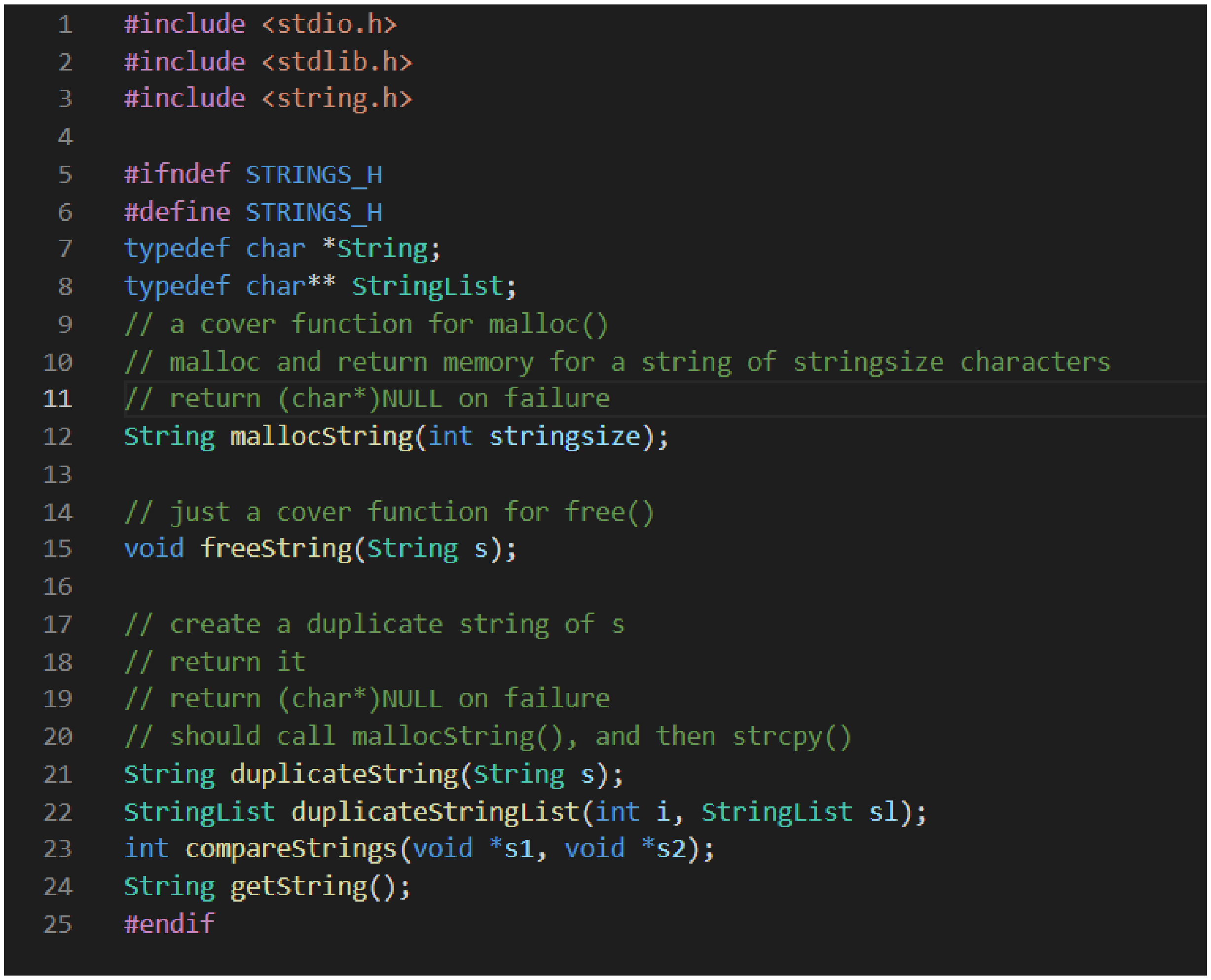


```c
#include "Strings.h"
#include <stdio.h>
#include <stdlib.h>
String mallocString(int stringsize)
{
    String pc = (String)malloc(sizeof(char) * (stringsize + 1));
    if (pc == (String)NULL)
    {
        return (String)NULL;
    }
    return pc;
}

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

String duplicateString(String s)
{
    String copy = mallocString(sizeof(s));
    if (copy == (String)NULL)
    {
        return (String)NULL;
    }
    strcpy(copy, s);
    return copy;
}
```



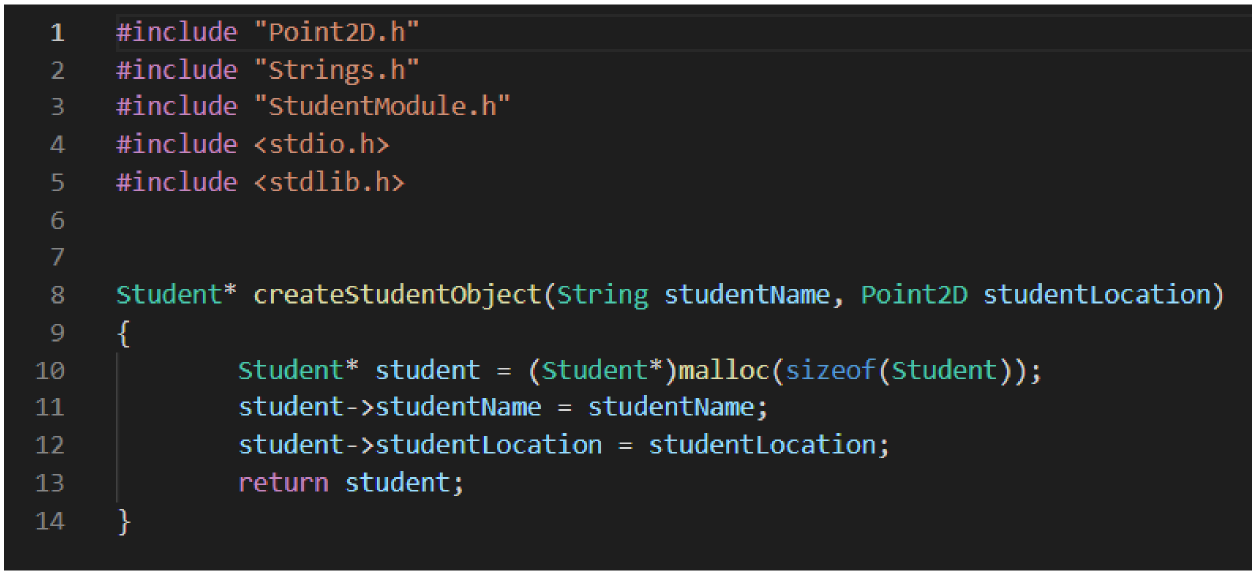
```c
StringList duplicateStringList(int i, StringList sl)
{
    StringList copy = (StringList )malloc(sizeof(String) * i);
    int j;
    for (j = 0; j < i; j++)
    {
        copy[j] = sl[j];
    }
    return copy;
}
int compareStrings(void *s1, void *s2)
{
    StringList sc1 = (String*)s1;
    StringList sc2 = (String*)s2;
    return strcmp(*sc1, *sc2);
}

String getString()
{
    String s;
    scanf("%[^\n]", s);
    return s;
}

```

Strings.h



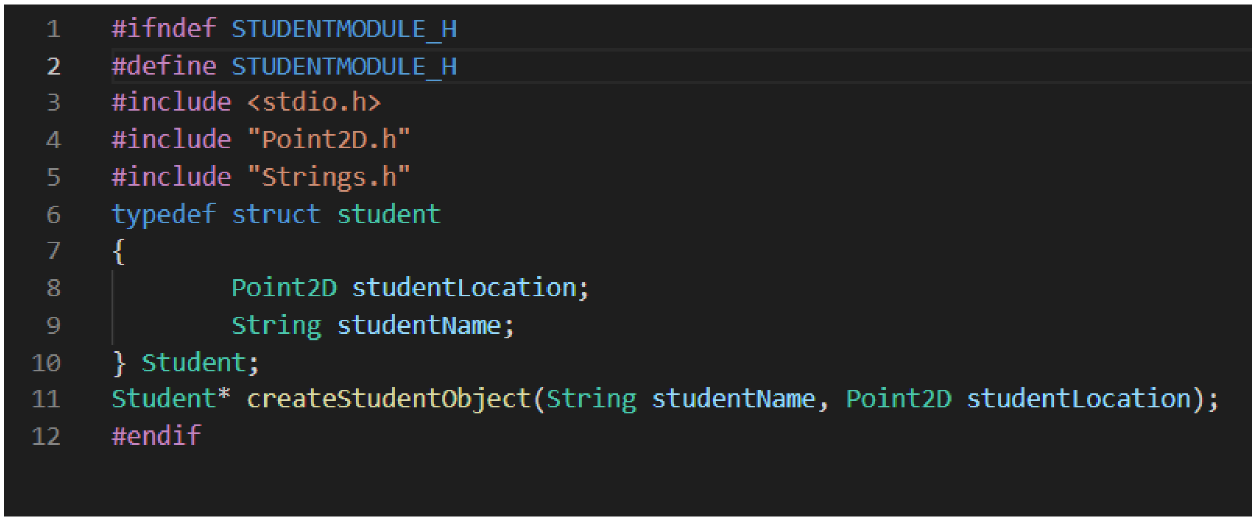
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifndef STRINGS_H
#define STRINGS_H
typedef char *String;
typedef char** StringList;
// a cover function for malloc()
// malloc and return memory for a string of stringsize characters
// return (char*)NULL on failure
String mallocString(int stringsize);

// just a cover function for free()
void freeString(String s);

// create a duplicate string of s
// return it
// return (char*)NULL on failure
// should call mallocString(), and then strcpy()
String duplicateString(String s);
StringList duplicateStringList(int i, StringList sl);
int compareStrings(void *s1, void *s2);
String getString();
#endif
```

StudentModule.c



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


Student* createStudentObject(String studentName, Point2D studentLocation)
{
        Student* student = (Student*)malloc(sizeof(Student));
        student->studentName = studentName;
        student->studentLocation = studentLocation;
        return student;
}
```

StudentModule.h



```c
#ifndef STUDENTMODULE_H
#define STUDENTMODULE_H
#include <stdio.h>
#include "Point2D.h"
#include "Strings.h"
typedef struct student
{
        Point2D studentLocation;
        String studentName;
} Student;
Student* createStudentObject(String studentName, Point2D studentLocation);
#endif
```