ForNextDay(16)
Stephen Cole
3553803

exercise.c

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

struct Point2D
{
	double x;
	double y;
};

void print(const struct Point2D* point)
{
	printf("%f,%f)\n", point->x, point->y);
}

int main(void)
{
	struct Point2D point = {.x = 1.1, .y = 2.2};
	print(&point);
	return 0;
}
```

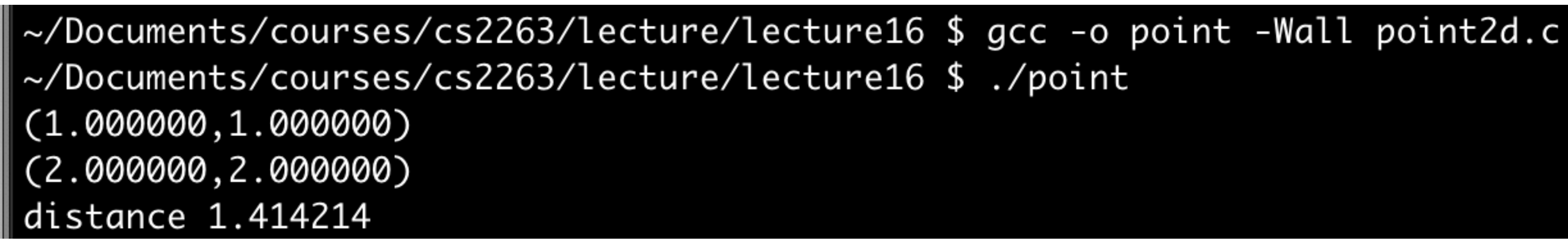
point2d.h

```
#ifndef POINT_2D_H
#define POINT_2D_H
typedef struct Point2D {
	double x;
	double y;
} Point2D;


void setPoint(Point2D* point, const double x, const double y);
void print(const Point2D* point);
double distance(const Point2D* point1, const Point2D* point2);

#endif
```

point2d.c

```
#include<math.h>
```

```
#include<stdio.h>

#include"point2d.h"

void setPoint(Point2D* point, const double x, const double y)
{
        point->x = x;
        point->y = y;
}

void print(const Point2D* point) {
        printf("(%f,%f)\n", point->x, point->y);
}

double distance(const Point2D* point1, const Point2D* point2) {
 const double x2 = pow(point2->x - point1->x, 2);
 const double y2 = pow(point2->y - point1->y, 2);
 return sqrt(x2 + y2);
}

int main(void)
{
        Point2D point;
        setPoint(&point, 1.0, 1.0);
        print(&point);
        Point2D point2;
        setPoint(&point2, 2.0, 2.0);
        print(&point2);
        printf("distance %f\n", distance(&point, &point2));
        return 0;
}
```



```
~/Documents/courses/cs2263/lecture/lecture16 $ gcc -o point -Wall point2d.c
~/Documents/courses/cs2263/lecture/lecture16 $ ./point
(1.000000,1.000000)
(2.000000,2.000000)
distance 1.414214
```

Strings.h

```
#ifndef STRINGS_H
#define STRINGS_H

typedef char* String;
```

```
// a cover function for malloc()
// malloc and return memory for a string of stringsize characters
// return (char*)NULL on failure
char* 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()
char* duplicateString(String s);

// create a duplicate of string list sl
// return it
// return (char**)NULL on failure
// uses other Strings module functions
char** duplicateStringList(String* sl,int n);

// Return an allocated string from an open file,
// Stop reading when any character is in terminators list
// return allocated string or (char*)NULL
char* getfString(FILE* pFIn, String terminators);
char* getString(String terminators);

#endif
```

Strings.c

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

String getfc(FILE* pFIn, String terminators, int n);
int StringInString(String t, String c);

// a cover function for malloc()
// malloc and return memory for a string of stringsize Stringacters
// return (String*)NULL on failure
String mallocString(int stringsize){
   return (String*)malloc(sizeof((stringsize+1)));
}
```

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

// create a duplicate string of s
// return it
// return (String*)NULL on failure
// should call mallocString(), and then strcpy()
String duplicateString(String s){
   String sCopy = mallocString(strlen(s));
   if(sCopy != (String*)NULL){
      strcpy(sCopy,s);
   }
   return sCopy;
}

String* duplicateStringList(String* s,int n){
   String* slCopy;
   // Allocate the list
   slCopy = (String**)malloc(sizeof(String*)*n);
   if(slCopy == (String**)NULL) return slCopy;

   // Allocate/duplicate the strings
   for(int i = 0; i<n; i++){
      slCopy[i] = duplicateString(s[i]);
      if(slCopy[i] == (String*)NULL){
         // Bad stuff - clean up and return
         for(int j=0; j<i; j++){
            freeString(slCopy[j]);
         }
         free(slCopy);
         slCopy = (String**)NULL;
         break;
      }
   }
   return slCopy;
}

// Return an allocated string from an open file,
// Stop reading when any Stringacter is in terminators list
// return allocated string or (String*)NULL
String getfString(FILE* pFIn, String terminators){
   String s = getfc(pFIn, terminators, 0);
   return s;
}
```

```
String getString(String terminators){
   String s;
   s = getfc(stdin, terminators, 0);
   return s;
}

String getfc(FILE* pFIn, String terminators, int n){

   String s;
   String c = fgetc(pFIn);
   //base case
   if(c == EOF || StringInString(terminators, c)){
      // allocate a string
      s = mallocString(n);
      if(s != (String*)NULL){
         // terminate the string
         s[n+1] = (String)NULL;
      }
      return s;
   }
   s = getfc(pFIn, terminators, n+1);
   s[n] = c;
   return s;
}

int StringInString(String t, String c){
   int i = 0;
   while(t[i] != (String)NULL){
      if(t[i] == c) return 1;
      i++;
   }
   return 0;
}

int compareStrings(String a, String b)
{
  int c = 0;

  while (*a == *b) {
    if (*a == '\0' || *b == '\0')
      break;
    a++;
                              b++;
  }
  if (*a == '\0' && *b == '\0')
    return 0;
  else
    return -1;
}

//End
```