

CS16 Final Exam

E03, 10W, Phill Conrad, UC Santa Barbara

Thursday, 03/18/2010

Name: _____

Umail Address: _____@ umail.ucsb.edu

Circle Lab section: 3PM 4PM 5PM

[Link to Printer Friendly PDF Version](#)

Please write your name **only** on this page. That allows me to grade your exams without knowing whose exam I am grading.

This exam is **closed book, closed notes, closed mouth, cell phone off**, except for:

- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes
- These sheets will be collected with the exam, and might not be returned
- Please write your name on your notes sheet

There are 100 points worth of questions on the exam, and you have 3 hours (180 minutes) to complete the exam.

A hint for allocating your time:

- if a question is worth 10 points, spend no more than 10 minutes on it
- if a question is worth 20 points, spend no more than 20 minutes on it
- etc.

That will leave you with 80 extra minutes to check your work, or come to problems that gave you trouble.

1. (5 pts) Write the definition of a C struct called `LatLng`—a struct that represents a latitude and longitude.

The struct should contain two members, both of type `double`: one called `lat`, and another called `lng` (note that you can't use the name `long` since `long` is a C language keyword.)

2. Assume that you are continuing with the same program in which you defined `struct LatLng` (from the previous question).

- a. (5 pts) Write the function prototype—for this question **only the prototype**—for a void function called `initLatLng` that takes three parameters:

- a. A parameter called `lPtr` that is a pointer to a `struct LatLng`
- b. A parameter called `theLat` of type `double`
- c. A parameter called `theLng` of type `double`

3. Still continuing with the same problem from questions 1 and 2:

- a. (10 pts) Now write the full function definition for `initLatLng`. The function should initialize the values in the struct pointed to by `ptr` with the values passed in in the two parameters, `theLat` and `theLng`.

Hint: I didn't leave much room for your answer, because the correct function definition is pretty short.

- b. (10 pts) Now, write a very short main program that does only three things inside the main:

- (1) Declares a variable of type `struct LatLng` with the name `sbaAirport`
- (2) Calls the `initLatLng()` function to initialize the variable `sbaAirport` to latitude 34.43, longitude -119.84
- (3) Returns a value to the operating system indicating "success".

Your program should have `#include "latLng.h"` before the main—you can assume that your definition of `struct LatLng` from question 1 and the function prototype for `initLatLng` both appear inside that header file.

(Note: Clearly, this program does no useful work—it would be only the starting point for a more useful program.)

4. (2 pts) When memory is allocated in a C program for a local variable in a function, is that memory allocated on the stack, or on the heap?

Circle one: stack heap

5. (2 pts) When memory is allocated in a C program by using malloc or calloc, is that memory allocated on the stack or on the heap?

Circle one: stack heap

6. (2 pts) As we discussed in lecture, recursive functions typically operate by calling themselves repeatedly until a base case is reached—with each call, a new copy of the function is created in memory.

Where are these copies placed in memory—on the stack, or the heap?

Circle one: stack heap

7. (3 pts) Suppose I have the variables:

```
char thisCity[20] = "Santa Barbara";  
char thatCity[20] = "Los Angeles";
```

If I want to set the contents of thatCity to be the same as the contents of thisCity, in C I can't just use:

```
thatCity = thisCity;
```

because that would change where thatCity points, and you can't change where the name of an array points in C.

What line of code do I write instead?

(Hint: you may assume that `#include <string.h>` appears at the top of the program.)

8. Below are two functions that find the sum of an array.

a. (3 pts) Is `sumArray1()` a recursive function? Why or why not? (Be specific)

b. (3 pts) Is `sumArray2()` a recursive function? Why or why not? (Be specific)

```
int sumArray1(int *a, int n)
{
    if (n<=0)
        return 0;
    else
        return a[0] + sumArray1(a + 1, n-1);
}

int sumArray2(int *a, int n)
{
    int sum=0;
    int i;
    for (i=0;i<n;i++)
        sum += a[i];
    return sum;
}
```

9. When I print out a pointer variable in the Ch interpreter or using the "%p" format specifier in printf, the output contains values such as these: 094b1e2c

a. (2 pts) What kind of value is this that contains both digits and letters such as—in this case—b, e and c?

b. (4 pts) Convert 094b1e2c to an equivalent representation in ones and zeros.

c. (2 pts) What does the numerical value of a pointer—a number such as 094b1e2c—actually mean?

Or to put it another way, how would you explain what a pointer really is—in plain english?

(I'm not looking for a long answer—it is possible to answer with only 1 or 2 words, or a very short phrase and get full credit—if your answer is accurate and to the point.)

10. Here is the definition for a C function called `copyPos()`—with a few parts missing that you need to supply.

The function `copyPos` should take an input array `src`, the length of the input array `src`, and an output array `dest`—the function copies only the positive numbers from `src` into `dest`, and returns the number of values copied into `dest`.

Fill in the missing lines of code in the places where you see comments with the `@@@` symbol

```
// Copy all strictly positive numbers---those greater than zero---
// from array src into array dest.
// n is the occupancy of array src.
// Destination array should have capacity at least of size n.
// Return the number of positive values copied into dest
```

```
int copyPos(int *src, int *dest, int n)
{
    int i;
    int count=0;

    for (i=0;i<n;i++)
    {
        if ( _____ > 0) // @@@ (4 pts)
        {
            _____ = src[i]; // @@@ (4 pts)
            count++;
        }
    }

    return count;
}
```

11. (5 pts) Here is a recursive function with the name `mysteryFunction`. As parameters, it takes an array of integers `a`, and the length of that array `n` as parameters, and returns an integer. What it actually computes is a mystery for you to solve.

```
int mysteryFunction(int *a, int n)
{
    if (n==0)
        return 0;

    if (a[0]%2==0)
        return a[0] + mysteryFunction(a+1, n-1);
    else
        return mysteryFunction(a+1, n-1);
}
```

Solve the mystery by figuring out what the mystery function computes, then decide which of the following is a more reasonable name for the function.

Among these possible function names, one (and only one) of them would be a reasonable answer to this question.

`average`, `countEven`, `countMax`, `countMin`, `countNeg`, `countOdd`, `countPos`, `countSevens`, `indexFirstEven`, `indexFirstOdd`, `indexOfMax`, `indexOfMin`, `isSorted`, `maxValue`, `minValue`, `noDups`, `sum`, `sumEven`, `sumNeg`, `sumOdd`, `sumPos`.

12. (12 pts) Together with this exam, there is a program (on a separate [handout](#)).

Assuming each of the expressions below appeared in this program, indicate the type they would have, or write error if the expression is not valid, e.g.

- dereferencing something with `*` or `->` that isn't a pointer
- a reference to a struct member that doesn't exist (e.g. `d.foo` where there is no `foo`)

The first few are done for you as an example.

Hints--for full credit:

- don't write *pointer to character*; instead, write **`char *`**
- don't write *address of int*; instead, write **`int *`**
- don't write *address of int ** or *address of pointer to int*, instead write **`int **`**

See [solution](#)

Expression	Type
<code>a</code>	<code>double *</code>
<code>*b</code>	error
<code>&e</code>	
<code>*f</code>	
<code>(*g).x</code>	
<code>h->y</code>	
<code>e->center</code>	
<code>j->m</code>	

13. (12 pts) On one of the homework assignments, you were given short sections of code involving pointers, and asked to make diagrams showing what the memory locations referred to look like after the code has executed.

On a separate handout given with this exam, there are a few examples of these problems, along with their solutions to remind you of how to make these diagrams.

Draw similar pictures to illustrate the variables, their values, and where the pointers point, for each of the following.

Give only the final values of the variables and pointers (at the point marked `//snapshot of memory taken here`)

Hint: Draw an initial diagram on scratch paper, or in the margin, then change the values and pointer arrows as the statements are executed. Then copy just the "final" version into the answer boxes.

Its ok if you go outside the boxes, as long as your answer is clear and legible.

code	picture
<pre>int a=2, b=1, *ptr=&a; b = *ptr; // snapshot of memory taken here</pre>	
<pre>int a=3, b=5, c=7, *ptr=&c; (*ptr) = a; a = b; b = (*ptr); // snapshot of memory taken here</pre>	
<pre>int a=3, b=5, c=7, *ptr1=&a, *ptr2=&b, *ptr3; ptr3 = ptr1; ptr1 = ptr2; ptr2 = ptr3; // snapshot of memory taken here</pre>	

14. (10 pts) On lab08, you were provided several definitions and function prototypes to work with. Here are a few of them:

```
struct Point {
    double x;

    double y;
};

void initPoint(struct Point *p, double xVal, double yVal);

struct Point makePoint(double xVal, double yVal);
void drawLine(struct Drawing *d, struct Point p1, struct Point p2, int color);
```

Using these, write the definition for a function drawH that draws the letter H (as shown below).

Here is the function prototype for the drawH function you should define:

```
void drawH(struct Drawing *d,
           struct Point ul, // upper left corner
           double w, // width
           double h, // height
           int color);
```

And here is a sample main, and its output

```
int main()
{
    struct Drawing d;

    initDrawing(&d, DRAWINGTYPE_COLOR, 200, 100, COLOR_WHITE);

    drawH(&d,makePoint(10,10),20,50,COLOR_RED);
    drawH(&d,makePoint(100,10),50,20,COLOR_BLUE);

    writeFile(&d, FILENAME);

    return 0;
}
```



If you need extra room, use the space on the next page

Extra space for your answer to question 14

End of Exam

Total Points: 100

CS16 Final Exam

Extra Handout

Side 1—for question 12

Program for question about types

```
// types.c Code for exam question, 11/15/2009
// P. Conrad for CS16, 09F, UCSB

#include <stdio.h>

struct Point {
    double x;
    double y;
};

struct Date {
    int d;
    int m;
    int y;
};

struct Circle {
    struct Point center;
    double radius;
};

int main(int argc, char *argv[])
{
    double *a;
    double b;
    int *c;
    int d;
    struct Circle *e;
    struct Circle f;
    struct Date *i;
    struct Date j;
    struct Point *g;
    struct Point h;

    // Program does no useful work
    // It is just the basis of a homework assignment about types

    // Pretend there is useful code here, and then
    // answer questions about the types of various expressions
    // as if they appeared right here.

    return 0;
}
```


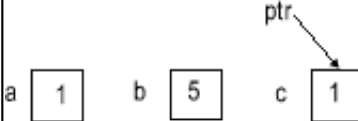
CS16 Final Exam

Extra Handout

Side 2—for question 13

Example diagrams to illustrate how to answer question 13

Note that in the second example, the value 1 shown in the picture for *c* is *not a typo*. This is because the line of code `(*ptr) = a;` changed that value from 5 to 1.

code	picture
<pre>int a=1, b=2, *ptr; ptr = &b; // snapshot of memory taken here</pre>	 <p>The diagram shows two memory locations, 'a' and 'b'. 'a' contains the value 1, and 'b' contains the value 2. A pointer labeled 'ptr' has an arrow pointing to the box containing '2'.</p>
<pre>int a=1, b=2, c=5, *ptr=&c; *ptr = a; // snapshot of memory taken here</pre>	 <p>The diagram shows three memory locations, 'a', 'b', and 'c'. 'a' contains the value 1, 'b' contains the value 5, and 'c' contains the value 1. A pointer labeled 'ptr' has an arrow pointing to the box containing '1'.</p>