Final Exam - Due 07/22/2020 @ 11:59 PM

**NAME: Todd Sipe**

## Basics

**Question 1**

Tell us the sizes of the following data types in C:

|  |  |
| --- | --- |
| Data Type | Size (in bytes) |
| char | 1 |
| short | 2 |
| long | 8 |

**Question 2**

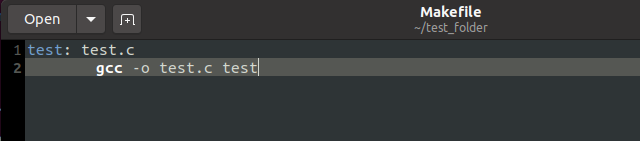
Say we have a c file called **test.c**, what is the Linux terminal command we would use to compile it and create an executable called **a.out**.

**Your Answer Here:**

**gcc test.c**

**Question 3**

Below is a screenshot of a Makefile I created, I made a syntax error, explain what I did wrong and how I can fix it.



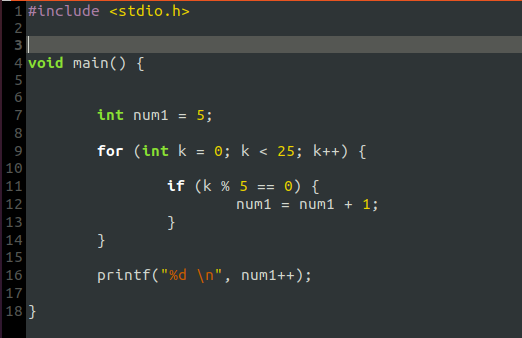
**Your Answer Here:**

Line 2 should be gcc -o test test.c

test and test.c were mixed up.

**Question 4**

Below is a screenshot of a c file called **do\_op.c**, what will the output be when I run this program? Use the space below to work through the code.



**Your Answer Here:**

10

## 

## Binary, Hex, and Bit Manipulation

**Question 1**

Convert the following decimal numbers into binary and hexadecimal: (I understand that you can just Google these conversions, but please do this on paper using the conversion techniques we learned! You **MUST SHOW ALL** work to get credit)

|  |  |  |
| --- | --- | --- |
| Decimal Number | Binary Representation (use 16-bits for each number) | Hexadecimal Representation |
| 18 | 10010 | 12 |
| 45 | 101101 | 2D |
| 800 | 110100000 | 320 |

**Decimal to Binary:**

Powers of 2 from 1024 down to 1: 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1

Subtract the highest possible power of 2 first, then repeat process until you hit 0. Whatever numbers you subtract are 1s, numbers you don’t subtract are 0s.

18 - 16 = 2, 2 - 2 = 0. 🡪 10010

45 – 32 = 13, 13 – 8 = 5, 5 – 4 = 1, 1 – 1 = 0. 🡪 101101

800 – 512 = 288, 288 – 256 = 32, 32 – 32 = 0. 🡪 110100000

**Decimal to Hexadecimal:**

Divide original number by 16. The remainder is the least significant digit. Repeat this process until you no longer get over 1, and use the hexadecimal representation for the remainders that you are left with. 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, and 15 = F.

18 / 16 = 1R2, 1 / 16 = 0R1 🡪 12

45 / 16 = 2R13, 2 / 16 = 0R2 🡪 2D

800 / 16 = 50R0, 50 / 16 = 3R2, 3 / 16 = 0R3 🡪 320

**Question 2**

In what kind of situation(s) would we want to use the hexadecimal number system?

**Your Answer Here:**

We would want to use hexadecimal for anything that we need to represent VERY large numbers in a small space. As we saw above, the decimal representation of 800 crunches all the way down to 320. If we had MANY digits in decimal form for whatever reason, it is represented with less digits in hexadecimal form because of it’s base 16 nature.

**Question 3**

Is it possible to multiply a number by 8 by only using a shifting operator? If so, explain how? (hint: think about this with binary numbers, write a few examples down below).

**Your Answer Here:**

Yes. Whatever number you have, convert it to binary, and use the binary representation of 8 and carry out the operation. Examples: 8 = 1000, 3 = 11, 6 = 110.

1000 1000

11 110

1000 0000

1000**0** 1000**0**

11000 1000**00**

110000

**Question 4**

Perform the following bitwise operations (steps 1-3):

1. XOR 21 (00010101) with 9 (00001001)
2. AND the result of **step 1** with 15 (00001111)
3. OR the result of **step 2** with 1 (00000001)

After these operations, what decimal number do we have?

**Your Answer Here:**

1: 00010101

00001001 XOR

00011100 = 28

2: 00011100

00001111 &&

00001100 = 12

3: 00001100

00000001 ||

00001101 = 13

## Functions

**Question 1**

Explain the difference between call by value and call by reference.

**Your Answer Here:**

Call by value uses the actual value of an argument, where the call by reference uses the address of the memory location of the argument that was stored.

**Question 2**

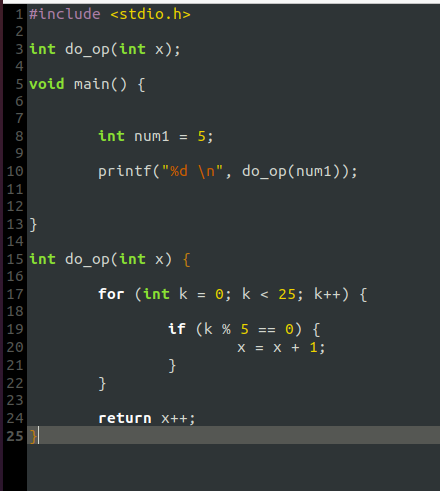
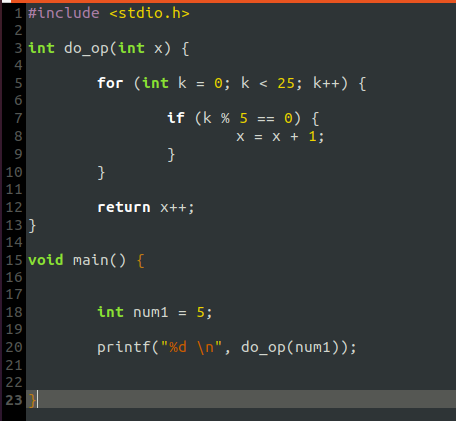
Is it possible to pass an array to a function using call by value? Explain why or why not.

**Your Answer Here:**

No, because an array is a series of values stored in a memory location. We would want to pass the address of the array into the function, rather than attempt to pass the array by a value.

**Question 3**

Below you will see two screenshots (option 1 and option 2), which of these is valid C code that will compile and run?



*\*Option 1 \*Option 2*

Choose the correct option here, delete the 2 options that are incorrect so that you are only left with one:

1. Both

## 

## Structs

**Question 1**

What is a struct and why is it useful?

**Your Answer Here:**

A struct is a set of data that can be molded and shaped any way by the programmer. It can be sets of particular data within other sets of particular data, within other sets, and so on. They are useful in compiling vast sets of information in a clearly defined way for the programmer or user to understand and access.

## 

## 

## 

## 

## 

## Pointers

**Question 1**

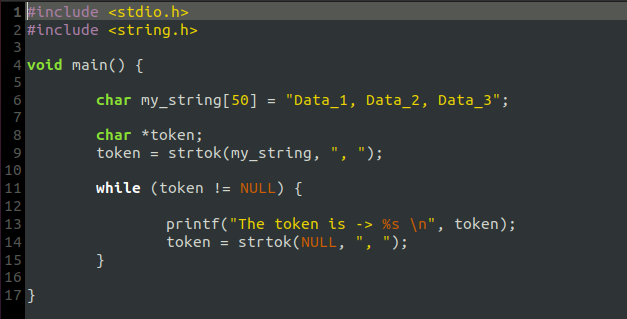
How can we use pointers to write more efficient code in terms of speed and memory usage? (hint: think about your lab 4 results)

**Your Answer Here:**

Pointers can be used to speed up the process of performing operations like moving, swapping, or whatever else. In our lab example we first swapped a hundred or so strings with each other many times over, and then we performed the exact same operations by using pointers to swap whatever is there. In my mind, the computer is manually swapping these strings character by character in the first instance, and in the second it’s saying “swap that one with that one”, rather than going character by character. This speeds up our process.

**Question 2**

Below you will see a screenshot of code that uses the **strtok** function, briefly explain how this function works. Your explanation must include details about how the function parses the data and how it uses a pointer to do this.



**Your Answer Here:**

This passes a pointer through the array of characters “Data\_1, Data\_2, Data\_3” starting at the very first character. What this is doing is it is looking for one of two things (for different reasons), either the NULL character (the end of the array of characters) or the specific set of characters “, ” which happens three times in the array. If it finds neither of these, it adds the character to the first token. If it finds the NULL character, it stops (because of the while loop). If it finds the “, ” characters, then it is finished with the creation of the previous token and begins creation of the next token.

The tokens will be:

Data\_1

Data\_2

Data\_3

## 

## Malloc

**Question 1**

In this class and on labs 6 & 7 we saw how we can use malloc to dynamically allocate space, explain what is meant by dynamically allocating space and how it is useful.

**Your Answer Here:**

If we stored something in a particular memory location, malloc allows us to reference the thing that we stored there. The process by which this is done is a little weird, but essentially we tell the computer to find this thing that we stored by giving it the address of the actual thing, which is stored elsewhere, so as to avoid an access violation. This allows us to dynamically fit our things into the spaces that we have allocated, that might not be big enough to store the things we wanted to store there initially.

This is really useful, because sometimes we will want to store something that is bigger than we anticipated it would be, or we come across some kind of situation where we will not know how big a particular thing is that we want to store in a piece of memory, so being able to dynamically allocate a new address for a larger thing using this method is pivotal.

## 

## 

## 

## 

## 

## Linked Lists

**Question 1**

**Part A.-** Name and explain 2 advantages that a linked list has over a traditional array.

1. We can adjust a linked list with less operations than we can with an array. To adjust some kind of value in the middle of an array we would have to probably change a lot of things in the array, where with a linked list we can just like bypass a node, or add a node in.
2. Linked list lengths can be expanded easily, by just adding nodes to the end, where arrays we would need to know the exact size before we got started.

**Part B.-** Name and explain 2 advantages that a traditional array has over a linked list

1. Arrays are a heck of a lot easier to write and compile. There is less chance for there to be an error that a junior level programmer would understand.
2. Arrays are probably a more efficient use of data storage, if you know the size initially of the array you need, you just allocate one small block of memory to the array, it would probably take more memory for the same amount of data to be stored in a linked list.