Pointers, Pointers and More Pointers!

LAB #07

SECTION #03

Jack Morrison

SUBMISSION DATE:

03/31/2023

Problem

The first problem is about getting aquainted with pointers. It asks you to simply print some different things about the pointer. Like the value stored at the memory address the pointer stores, or the memory address the pointer stores, or the memory address of the pointer itself.

The second problem is mostly about malloc. It asks us to create a program that takes in user input. The input is a users name and major. However, we must use malloc to create pointers to allocated memory in order to store this data.

The third problem is getting used to how pointers work, especially pass-by-pointer. It asks us to create a function that takes an input price and amount paid in pennies, as well pointers to outputs in dollars, quarters, dimes, nickels, and pennies.

The bonus problem asks us to dynamically allocate memory for an array which holds a variable number of strings which are the names of students. We must first take an input for the number of students then inputs for each student name.

Analysis

This first problem is three simple print statements like *ptr, ptr, and &ptr. Each of these will do different things.

The second problem we have to allocate using malloc and sizeof to get the size of a type. So we have to analyze how much storage space we need. We must also remember to free that memory after we are done with it.

The third problem has us passing data by pointer instead of by value. We have to use a function to calculate the change left over from a transaction. So there is a constraint on how we must design the code because we must use pass by reference.

In the bonus question we must take input and then allocate array space, then loop and allocate string space. This is because the problem is asking us to store everything in memory dynamically.

Design

Not much design for the first problem just three print statements. But each print statement does something different. The first dereferences the pointer and prints the value stored in the address that the pointer points to. The second prints the value of the pointer which is the address that the pointer points to. And the third prints the value of a new pointer which contains the address of the original pointer. Pointers are usefull for many things but often are used for mutating memory in places where they don’y have access to the memory directly. This is especially clear with malloc, because malloc allocates space on the heap for your data, which as a programmer you don’t have to worry about most of the time.

Fot problem two the hardest part is getting the argument to malloc correct. To store a string you’ll want to use sizeof(char) and multiply it by the number of chars you want to store. You always should free memory that you allocate with malloc because if you don’t the allocator will be unable

to reclaim that space for other things. This is a bug known as a memory leak it is when your program uses a lot of memory storing things that it no longer keeps references to or cares about. This can cause several issues usually it slows down your computer if it is bad enough.

For problem three I designed one helper function for the main makeChange function to use. It is a function I named getRemaining it takes two inputs and one output pointer. It uses the two inputs to calculate the amount remaining after dividing using modulus. The function returns the integer amount you can divide by. I can then use it to calculate the number of dollars, quarters, or any amount of change in a given number of pennies. For example getting the number of dollars in pennies would look like:

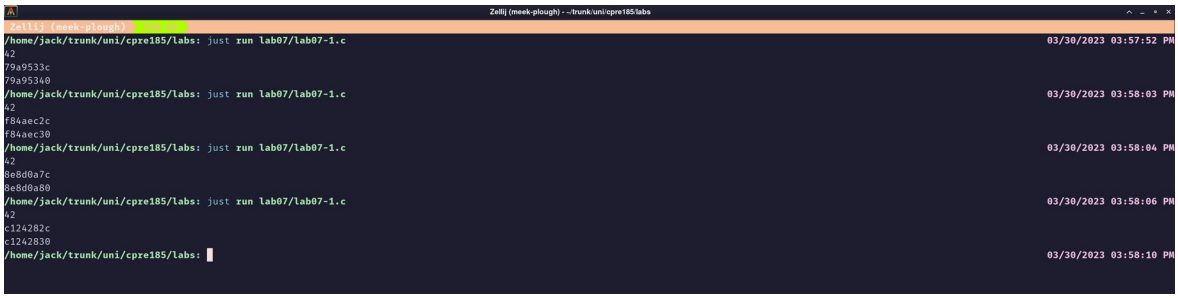
```
int dollars = getRemaining(100, pennies, &remainder);
```

For the bonus problem you have to create a two dimensional array in memory so to do that you have to malloc space for char pointers. You can do that with sizeof(char*) then you multiply by the number of students entered to get the desired length. Then you just have to loop over the input and malloc space for regular strings then once they are entered you can realloc them to trim them down to a minimal size. But if you trim them down you have to remember to use strlen() + 1 for the allocation size because of the null byte. Then afterwards you can do whatever you want with them, however to free them you must first loop over the array and free each index before freeing the array.

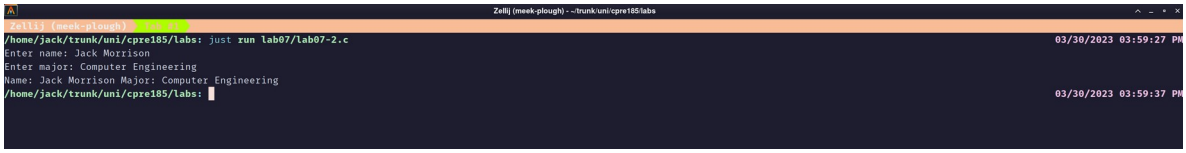
Testing

I already knew how pointers worked for problem one and two so I didn't do really any testing of those programs, I just verfied that what I knew was correct. For problem 3 I initially had seperate functions for each remainder amount but I factored it into a single function for ease of use. For the bonus problem I just tested some minor additions to the main function like a loop that prints the students that you entered.

Screen Shots



Problem 1



Problem 2

Problem 3

Bonus Problem