Pointers

What's a Pointer

A pointer is something that “points to” something that lives in memory.

Well, how do you “point to” memory? In a computer, anything in memory has an address. The address is just a number, like a street address. In C, you denote a pointer by using a “*”, (read as “star”). So, an int* is an “int star” or “pointer to an int”.

Pointer Operators

There are two main operators that are specific to pointers as well as arithmetic operators.

Ampersand

The ampersand means “take the address of”. Remember, a pointer is an address. Therefore, the ampersand operator gives back a pointer.

```
int a; //Just an int
&a; //int* (a pointer to an int)
&&a; //int** (a pointer to a pointer to an int)
```

This is called ”referencing” something.

Star

When not used with a type (eg int*), than the '*' operator is just the opposite of the '&'. The '&' takes the VALUE that a pointer points to. So, '*' and '&' are inverse operatons.

```
int** point; //An int** (a pointer to a pointer to an int)
*point; //int*
**point; //int
```

This is called “de-referencing” something.

Arrays

When you have an array variable in C, you don't actually have all the elements in that array. You actually just have a pointer to the first elements in that array. This is why you can denote arrays as 'int*' or 'char*'.

```
int a[]; //Array if ints
int* b; //Array of ints OR just a pointer to an int
int* c[]; //A pointer to an array of ints
int** d; //Either a pointer to an array of ints or a pointer to a pointer to an int
```

When you have a an array, you can use arithmetic operations on the pointer. Looks at these two pieces of code:

```
int nums[5] = {1, 2, 3,
4, 5};
int i;

for (i = 0; i < 5; i++)
{
   printf("%d\n",
nums[i]);
}
```

```
int nums[5] = {1, 2,
3, 4, 5};
int i;

for (i = 0; i < 5; i+
+)
{
   printf("%d\n",
*(nums + i));
}
```

These pieces of code do the exact same thing, they print every element in nums. You have seen the first one many times, look closely at the print in the second example. This line especially:

```
printf("%d\n", *(nums + i));
```

What does `*(nums + i)` do? When we add to a pointer, we get a pointer to the next element in the array. Then, we use a '*' on the pointer to de-reference it and we get an int back.

Technical Explanation

When you add an integer to a pointer, the compiler knows that you are trying to add to a pointer and instead of adding that integer number, it adds that integer number multiplied by the size of whatever the pointer points to. This is so that if you have an array and you have a pointer to the first element in that array, you can add one to that pointer and you will now be pointer to the next element.

Pointer v Array v String

A pointer is an address to something that resides in memory.

An Array is a contiguous set of locations in memory. Arrays are usually stored as just a pointer to the first element in the array.

A string is a character array that ends with a '\0'.

Out-Parameters

Out parameters (“out params”) are one of the most useful things that you can do with pointers. Recall that when you pass something to a function, the computer makes a copy of whatever you pass and gives the copy to the function. So, if you change a parameter, it is not changed outside of the function. Also recall that functions can only return one thing.

By using out parameters, you can change things outside of your function and effectively return multiple things.

You do this by passing a pointer as a parameter. The function then gets a copy of that pointer, but they both still point to the same location in memory. So if you de-reference the pointer, you can get change that the pointer is pointing to.

```
void outParamFun();

int main()
{
   int num = 5;

   printf("%d\n", num); //Prints "5"
   outParamFun(&num);
   printf("%d\n", num); //Prints "-11"

   return 0;
}

void outParamFun(int* numPoint)
{
   *numPoint = -11;
}
```

Note that in the example nothing is returned from the function, and when we call it we use &num.

Program

For this program, I want you to write me a program that will find the min, max, and average of an array of integers. You can either accept the array from the user, or just hard code it into your program.

Specs

1. Somehow get an array of 5 ints (read in or hard code).
2. Make a function that takes the following:
    1. An array of ints
    2. The size of the array of ints
    3. A pointer to an int
        1. This will be the out param for the minimum
    4. A pointer to an int
        1. This will be the out param for the maximum
    5. A pointer to an double
        1. This will be the out param for the average value in the array
3. In this program, you are not allowed to use array notation ('[]') when accessing an array.
    1. You can still use the brackets to declare you array.
    2. "`int nums[5]`" is OK
    3. "`nums[0] = 3`" in NOT ok.
4. You can assume that the array will always have at least one element.

Functions

1. `int main();`
2. `void numStats(int* nums, int size, int* min, int* max, double* avg);`
    1. Must calculate the max, min, and average of the array and return as out params.

Test Runs (input in **bold**)

```
Please enter 5 ints: 123 1255 5685 346346 8962
Min: 123, Max: 346346, Avg: 72474.200000

Please enter 5 ints: 1 2 3 4 5
Min: 1, Max: 5, Avg: 3.000000
```