CSCD 240

NOTE: Your answers, for all problems, will be saved in a file named cscd240Lab6pointers.pdf for all problems

NOTE: Your C file will be named cscd240Lab6.c

1) Type in, compile and execute the following code.

```
#include <stdio.h>
int main()
{
        int arr[] = { 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000};
        int *ptr = arr;

        /* This gives us an idea of the memory map */
        printf("sizeof(ptr) %ld\n", sizeof(ptr) );
        printf("sizeof(arr[0]) %ld\n", sizeof(arr[0]) );

        printf("arr %p\n", arr);
        printf("ptr %p\n", ptr);

        printf("arr[1] %p\n", &arr[1]);
        printf("arr[9] %p\n", &arr[9]);
        printf("&ptr %p\n", &ptr);
        /* end memory map */
        return 0;

}// end main
```

This code will provide a base address for arr and ptr.

For this problem use the base address of 0x5600bc

Answer /complete the following

a) What is the size of ptr?
   The size of ptr is 8.
b) What is the size of arr[0]?
   The size of arr[0] is 4.

c) Draw a memory map that shows the memory locations of each element of the array and of ptr. Please follow the patterns that we used in the classroom, with values shown in memory cell (box) and address shown outside of memory cell.

Base Address: 0x5600bc
Arr[0] = 0x5600c0
Arr[1] = 0x5600c4
Arr[2] = 0x5600c8
Arr[3] = 0x0560cc
Arr[4] = 0x0560d0
Arr[5] = 0x0560d4
Arr[6] = 0x0560d8
Arr[7] = 0x0560dc
Arr[8] = 0x0560f0
Arr[9] = 0x0560f4
*ptr = 0x0560c0

2) Use the base address provided from problem #1 as the base address of the array. Based on the code below, create an educated guess that clearly outlines what you believe will happen as each line is executed. In your explanation clearly **explain what is happening**, don't just give memory addresses or values. If you only provide memory addresses or values you will receive **0 points** for this problem. Your guesses will be clearly labeled in the PDF file. You must provide the line of code and then the explanation. NOTE: Where I provide the comment // reset ptr no explanation is required.

Note: the code provided in the below is code segment, not a complete program.

```
Ptr++; //I believe in this line that ptr increases increment up to arr[1] thus taking arr[1] memory
        //location.
printf("*ptr %i\n", *ptr); //this line prints the value of arr[1]
printf("ptr %p\n", ptr); //this line prints the memory address of &arr[1]

*++ptr; //This increases the value of ptr from arr[1] to arr[2]
printf("*++ptr %i\n", *ptr); //this line prints the value of arr[2]
printf("ptr %p\n", ptr); //this line prints the memory address of &arr[2]

*ptr++; //This line increases the value ptr from arr[2] to arr[3]
printf("*ptr++ %i\n", *ptr); //this line prints the value of arr[3]
printf("ptr %p\n", ptr); //this line prints the memory address of arr[3]
```

```
ptr = arr; // reset ptr

// fun with printf repeat last couple of commands
printf("*++ptr %i\n", *++ptr); //line prints the value of arr[1], while adding the 1 increment each time executed, preincrement.
printf("ptr %p\n", ptr); //line prints the memory address of arr[1]

printf("*ptr++ %i\n", *ptr++); //line prints the value of arr[1], while adding 1 increment each time executed post increcrement.
printf("ptr %p\n", ptr); //line prints the memory address of arr[1]

ptr = arr; // reset ptr

*ptr += 1; //adds +1 to the arr[0] value
printf("*ptr %i\n", *ptr); //prints out 201, as the value of arr[0]
printf("ptr %p\n", ptr); //prints out the memory location of arr[0]

printf("*(ptr+1) %i\n", *(ptr+1)); //increments the value of ptr by 1 resulting in it's value being that
                                   //of arr[1] being 400

ptr = arr; // reset ptr


*(arr+2) = *ptr+100; //adds 100 to the value of arr[0]
printf("*(arr+2) %i\n", *(arr+2)); //prints out the value of arr[0]

ptr = arr + 5; //adds five to the value of arr resulting in the value of arr being arr[6]
printf("*ptr %i\n", *ptr); //prints out the value of arr[6]
printf("ptr %p\n", ptr); //prints out the memory location of arr[6]

ptr = arr; // reset ptr

arr[2] = *(ptr + 5);  //sets the ptr value of arr[2] to that of arr[6]
printf("arr[2] %i\n", arr[2]); //prints out the value of arr[2] which is now arr[6]

ptr = (arr + 10); //sets the ptr value to that of arr[10]
printf("ptr %p\n", ptr); //prints out the value of arr[10]
printf("*ptr %i\n", *ptr); //prints out the memory location of arr[10]
```

3) Edit the C file
   a) Add the code from problem #2 to your C file
   b) Compile and execute your C file - capture the output
   c) In the PDF clearly state the line of code, your guess and what the result was. If you guessed correctly then state – correct guess, otherwise clearly explain the incorrect guess.

Guess	Line of Code
Correct	21
Correct	25
Correct	29
Correct	36
Correct	37
Correct	44
Correct	48
Correct	53
Correct	54
Correct	56
Correct	57
Correct	58

   d) Explain how the value for *ptr was deteremined based on

```
ptr = (arr + 10);
printf("ptr %p\n", ptr);
printf("*ptr %i\n", *ptr);
```

The value of ptr was determined based on this line due to the fact that we are giving ptr the value of arr + 10. Thus changing the value of ptr from arr to arr+10 resulting in the change of the ptr value. Note that arr[10] doesn't exist so I'm assuming that the segment of code picked a random piece in memory.

TO TURN IN:

A zip file containing:

- Your PDF file
- Your C file

You better know the naming scheme.