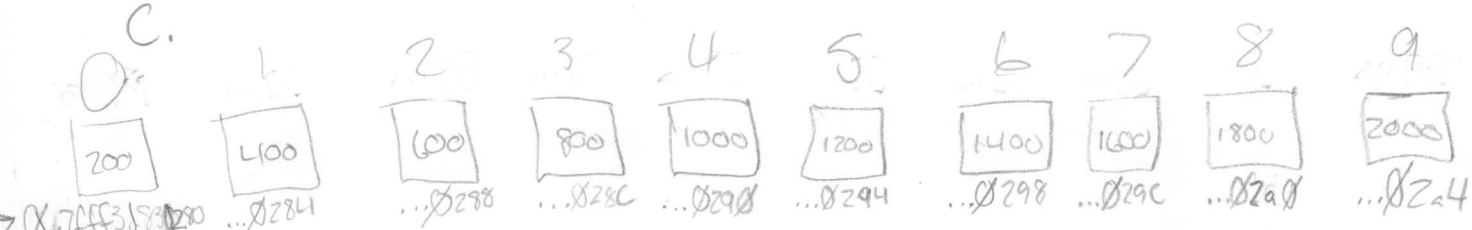
(1) a. 8

b. 4

c.



(2) int *ptr

ptr++; → address of arr[1]

printf("*ptr%

2) Use the provided address from #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 the type is, what the value would be and 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.

Example: printf("ptr %?\n", ptr) -> %p - int pointer the value is 0xffccd0 – printing the results from the previous line.

ptr++; → int *ptr → address of arr[1]
printf("*ptr %?\n", *ptr); prints value 400 at address of arr[1]
printf("ptr %?\n", ptr); prints address of arr[1] → 0x7fff3d83d284

// we talked about why we hate this but you need to understand
*++ptr; → int *ptr +1 int to address → now address of arr[2] → ...d288
printf("*++ptr %?\n", *ptr); prints value 600, value at address of arr[2]
printf("ptr %?\n", ptr); prints address of arr[2] → ...d288

*ptr++; → int *ptr +1 int to address → now address of arr[3] → ...d28C
printf("*ptr++ %?\n", *ptr); prints value 800, value at address of arr[3]
printf("ptr %?\n", ptr); prints address of arr[3] → ...d28C

ptr = arr; // reset ptr

// fun with printf repeat last couple of commands
printf("*++ptr %?\n", *++ptr); prints value 400, value of address arr[0] +1 int → arr[1]
printf("ptr %?\n", ptr); prints address

printf("*ptr++ %?\n", *ptr++); prints value 400 → happens before adding 1 int to the address
printf("ptr %?\n", ptr); prints address of arr[2] → ...d288

ptr = arr; // reset ptr

*ptr += 1; adds 1 to the value at the address arr[2] → 600 +1 → 601
printf("*ptr %?\n", *ptr); prints the value 601 → value at address of arr[2]
printf("ptr %?\n", ptr); prints the address of arr[2] → ...d288

```
printf("*(ptr+1) %?\n", *(ptr+1)); add 1 int to the address, print that value → 800
ptr = arr; // reset ptr

*(arr+2) = *ptr+100;
printf("*(arr+2) %i?n", *(arr+2));

ptr = arr + 5;
printf("*ptr %?\n", *ptr);
printf("ptr %?\n", ptr);

ptr = arr; // reset ptr

arr[2] = *(ptr + 5);
printf("arr[2] %?\n", arr[2]);

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

3) Edit the C file
   a) Add the code from problem #2 to your C file change the %? to the appropriate conversion character – either %d or %p
   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.
   d) Explain how the value for *ptr was determined based on

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

TO TURN IN:

A zip file containing:

- Your PDF file
- Your C file

The zip will be named your last name first letter of your first name lab8.zip (Example: steinerslab8.zip)

```
Blakes-MacBook-Air:lab8 Blake$ ./a.out
sizeof(ptr) 8
sizeof(arr[0]) 4
arr 0x7fff586dabb0
ptr 0x7fff586dabb0
arr[1] 0x7fff586dabb4
arr[9] 0x7fff586dabd4
&ptr 0x7fff586daba0
*ptr 400
ptr 0x7fff586dabb4
*++ptr 400
ptr 0x7fff586dabb4
*ptr++ 400
ptr 0x7fff586dabb4
ptr 0x7fff586dabb0
*ptr++ 200
ptr 0x7fff586dabb4
*ptr 201
ptr 0x7fff586dabb0
*(ptr+1) 400
*(arr+2) 301
*ptr 1200
ptr 0x7fff586dabc4
arr[2] 1200
ptr 0x7fff586dabd8
*ptr 196774900
```

d. the value for pointer was determined by getting the address of arr and adding 10 ints to it (10 x 4 = 40) you will have to use the correct hex values