280

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
arr -->	200	400	600	800	1000	1200	1400	1600	1800	2000

2a8
ptr

1.

a. 8 bytes
b. 4 bytes

2

ptr++;
increments the pointer by 1 int, or 4 bytes

printf("*ptr %i\n", *ptr);
will print out the second element of the array
correct guess

printf("ptr %p\n", ptr);
will print out memory location of the second element of the array
correct guess

ptr = arr; // reset ptr

*++ptr;
increments the value of the second element of the array
incorrect guess. The right answer was it will give you the value of the second element of the array

printf("*++ptr %i\n", *ptr);
will print out 401, as the value was incremented, not the pointer itself
incorrect guess. Correct answer was 600

printf("ptr %p\n", ptr);
will print out the memory location of the second element of the array
incorrect guess, the answer was the memory location of the second element of the array

ptr = arr; // reset ptr

*ptr++;
increments the value of the second element in the array
incorrect guess. Correct answer was it would point at the third element of the array

printf("*ptr++ %i\n", *ptr);
will print out 402, since the value was incremented
incorrect guess

printf("ptr %p\n", ptr);
will print out the memory location for the second element in the array
incorrect guess

ptr = arr; // reset ptr

printf("*++ptr %i\n", *++ptr);
will print out 201 b/c the value is getting incremented
incorrect guess

printf("ptr %p\n", ptr);
will print out the memory location of the first element in the array
incorrect guess

printf("*ptr++ %i\n", *ptr++);
will print out 202 b/c the value is getting incremented.
Incorrect guess

printf("ptr %p\n", ptr);
will print out the memory location of the first element in the array
incorrect guess

ptr = arr; // reset ptr

*ptr += 1;
incrementing the first value in the array by 1
incorrect

printf("*ptr %i\n", *ptr);
will print out the value of the first element in the array
incorrect

printf("ptr %p\n", ptr);
will print out the memory location of the first element of the array
incorrect

printf("*(ptr+1) %i\n", *(ptr+1));
will print out the second element of the array
incorrect

ptr = arr; // reset ptr

*(arr+2) = *ptr+100;
will assign the third element of the array with the value of the first element plus 100
incorrect

printf("*(arr+2) %i\n", *(arr+2));
will print out the value of the third element of the array
incorrect

ptr = arr + 5;
it will increment the memory location at ptr by 20 bytes, or 5 spots forward in the array

printf("*ptr %i\n", *ptr);
will print out the value of the sixth element of the array

printf("ptr %p\n", ptr);
will print out the memory location of the sixth element of the array

ptr = arr; // reset ptr

arr[2] = *(ptr + 5);
will assign the third element of the array the value of the sixth element of the array
correct

printf("arr[2] %i\n", arr[2]);
will print out the value at the third element of the array
correct

ptr = (arr + 10);
pointer is pointing off the list, as the list isn’t bigger than 10 elements
correct

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

will print out the memory location that ptr is currently at (i.e. not on the list)
correct

printf("*ptr %i\n", *ptr);
will print garbage values
correct

3.

a. did it
b.

```
sizeof(ptr) 8
sizeof(arr[0]) 4
arr 0x7ffe43995a10
ptr 0x7ffe43995a10
arr[1] 0x7ffe43995a14
arr[9] 0x7ffe43995a34
&ptr 0x7ffe43995a08

*ptr 400
ptr 0x7ffe43995a14
*++ptr 600
ptr 0x7ffe43995a18
*ptr++ 800
ptr 0x7ffe43995a1c
*++ptr 400
ptr 0x7ffe43995a14
*ptr++ 400
ptr 0x7ffe43995a18
*ptr 201
ptr 0x7ffe43995a10
*(ptr+1) 400
*(arr+2) 301
*ptr 1200
ptr 0x7ffe43995a24
arr[2] 1200
ptr 0x7ffe43995a38
*ptr 616014592
```

c.