CS221
C and Systems
Programming

More fun with pointers and `malloc()`

Several ways to iterate over an array:

```
int* ptr = malloc(sizeof(int) * LENGTH);
for(int indx = 0; indx < LENGTH; indx++) {
 ptr[indx] = value(); // first
 *(ptr + indx) = value(); // second
 *ptr++ = value(); // third
}
```

What is "ptr + indx"?
Walk thru all three
- See how the third breaks the first two?

How does that work?

```
int* iPtr = …
iPtr += 1;


char* cPtr = …
cPtr += 1;
```

How much does pointer move in each case? 1 "unit", not necessarily 1 byte.
LOOK AT pointerFun.c

Then cast the pointers to char*s and see what you get. Then double*s.

Ugh! Why are we doing this?

```
void mergesort(int* input, int length) {
 if (length >= 2) {
    mergesort(input, length/2);
    mergesort(input+length/2, length/2);
 }

 // Merge the two half-sized arrays
 …
}
```

4

We can create a NEW ARRAY from the middle of an existing array. Must copy the array Java.
We can create methods that work with arrays and very easily hand them subarrays
Super-common in video processing.

Bad pointers

Bad pointers?

Segmentation violation

```
char* myString = malloc(64);
myString[-8000] = 'A';

char* myString = "string stored in Constant Memory Space";
myString[0] = 'S';
```

Access an illegal/inaccessible address.
Many "incorrect" addresses do not cause a segmentation violation, just a bug.

Bad pointers?

Bus error
```
char* myString = malloc(1000);
int* someIntegers = (int*) &(myString[1]);
```
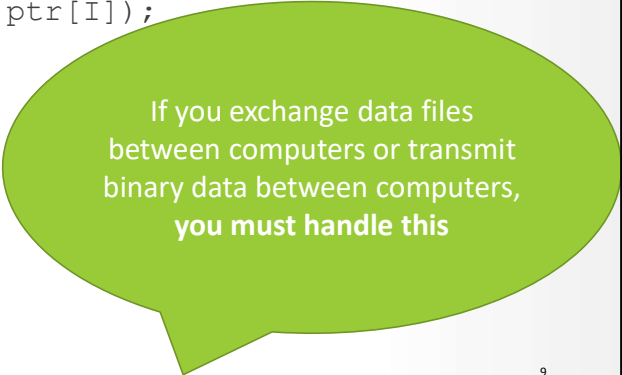
Modern computers may not declare an error.

Other types of hardware may declare an error.

7

What is the error? INT is not aligned on 4-byte boundary.

7

Ordering and "endianness"

endianness

```
int value = 0x0a0b0c0d;

char* ptr = (char*) &value;

for (int I = 0; I < 4; I++) {
  printf("%lx %x\n", ptr + I, ptr[I]);

}
```

If you exchange data files between computers or transmit binary data between computers, **you must handle this**

9

Do a sample program. Little endian.

How about shorts? Doubles? Floats?