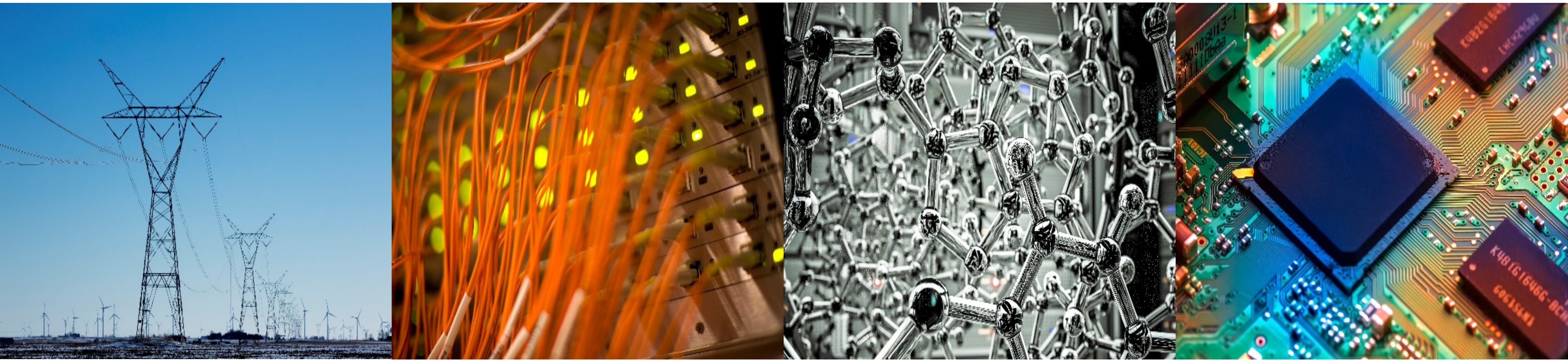
ECE 220 Computer Systems & Programming

Lecture 25 – Dynamic Memory Allocation

July 21, 2020



ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

- Schedule MT2 with CBTF

Lecture 24 Recap

“Static” vs. Dynamic Memory Allocation

- mechanism
- lifetime
- location
- size

Memory Leak vs. Segmentation Fault

Exercise:

```
typedef struct studentStruct{
   char *NAME;
   int UIN;
   float GPA;
}student;
```

1. Dynamically allocate memory for 200 student records (**hint: you will also need to allocate an array of 100 chars to hold the name for each record**)
2. Initialize name to “To be set”, UIN to -1 and GPA to 0.0 for all 200 records

```
void *malloc(size_t size);
void free(void *ptr);
```

calloc & realloc

```
void *calloc(size_t n_items, size_t item_size);
```

- similar to malloc(), also sets allocated memory to zero
- n_item: the number of items to be allocated, item_size: the size of each item → total size of allocated memory = n_items * item_size

```
void *realloc(void *ptr, size_t size);
```

- reallocate memory block to a different size (change the size of memory block pointed to by ptr)
- returns a pointer to the newly allocated memory block (it may be changed)
- Unless ptr == NULL, it must be returned by the malloc() family of functions
- if ptr == NULL → same as malloc()
- if size == 0, ptr != NULL → same as free()

Example using calloc & realloc

- What does this block of code do?

```
char *ptr2 = calloc(100, sizeof(char));
if(ptr2 == NULL){
   printf(“ERROR – calloc failure!”);
   return 1;}
strncpy(ptr2, “Example using calloc”, 100);
```

- What is happening now?

```
char *ptr3 = realloc(ptr2, 200*sizeof(char));
if(ptr3 == NULL){
   printf(“ERROR – realloc failure!”);
   return 1;}
```

- How many bytes are we deallocating here?

```
free(ptr3);
```

Exercise:

```
typedef struct studentStruct{
   char *NAME;
   int UIN;
   float GPA;
}student;
```

1. Dynamically allocate memory for 200 student records (**hint: you will also need to allocate an array of 100 chars to hold the name for each record**)
2. Initialize name to “To be set”, UIN to -1 and GPA to 0.0 for all 200 records
3. Add 200 more student records and initialize them as in step 2
4. Free up memory space for all the records