8/27/21 COP 3502

1. UCF Programming Team
   - Announcement WebCourses
2. Dynamic Memory Allocation
3. Office Hours for TAs Posted
4. PO - Due Tonight
   Late Due Wednesday 9/1

Statically Allocated Memory

All you've ever used...

(1) Size known at compile time

eg
```
int x;
char str[100];
```

(2) Scope - is always within the function it's declared and within the nearest curly braces.

eg
```
for (int i=0; i<100; i++) {
    char str[100];

} //str dies here!
```

(3) from the stack (reasonably limited)

If you only used statically allocated memory in C you can't:

(1) Declare a large array.

(2) Create memory in a function and have it persist after the function is over.

Dynamically Allocated Mem

1. Can allocate large amounts
2. Can persist after func over
3. Because this memory doesn't get freed at the end of a function, YOU have to free it:
   - "Each and every malloc has an equal and opposite free"
   - "Clean up after yourself!"

stdlib.h

```
void* malloc(int size)
```

it finds size # of bytes and returns a pointer to it. If the memory wasn't found NULL is returned.

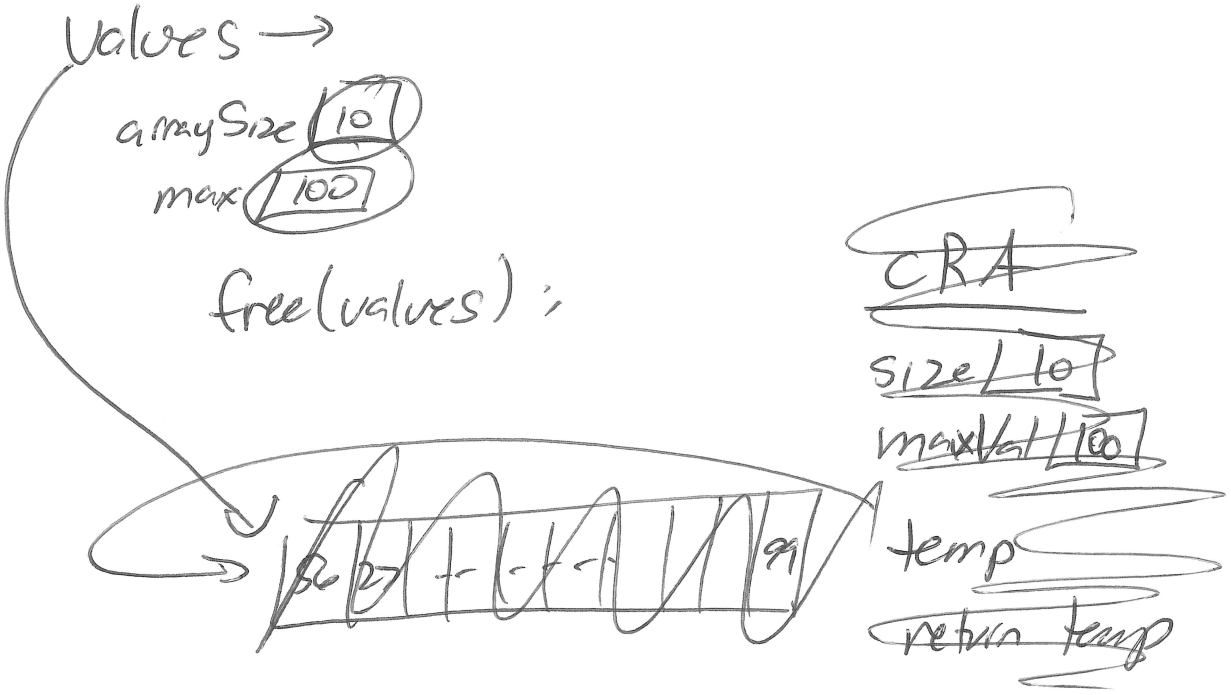
n 52

```
int* array = malloc(n * sizeof(int));
```

Dyn Mem Pic

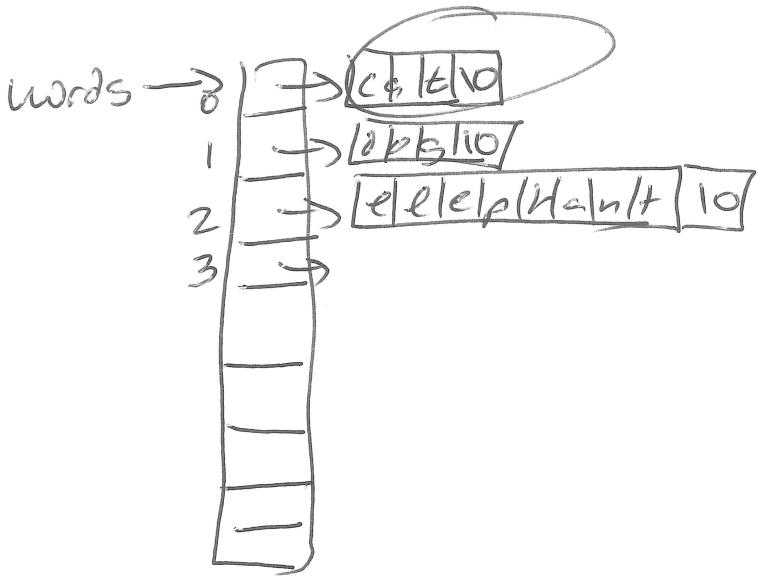
main

int* values = cRA(arraySize, max)



Dictionary of words

2D array



1st malloc

```
char** words = malloc(nWords * sizeof(char*));
for (i=0; i<nWords; i++)
    char temp[100];
    scanf("%s", temp); // big enough
    words[i] = malloc((strlen(temp)+1) *
                      sizeof(char));
}
```

To free

```
for (i=0; i<nWords; i++)
    free(words[i]);
free(words);
```