CS 161
Intro to CS I

Static vs. Dynamic Arrays
Odds and Ends...

• Assignment #4 due tonight
• Assignment #5 posted
```cpp
#include <iostream>

using namespace std;

#define NUM_ELEM 4 // make a macro or constant for number of elements

int main() {
    int a[NUM_ELEM] = {1}; // will only initialize first element to 1
    for (int i = 0; i < NUM_ELEM; i++)
        cout << a[i] << endl; // print initial values of elements
    for (int i = 0; i < NUM_ELEM; i++)
        a[i] = 2; // initialize all elements to 2
    cout << a << endl; // print address of where array begins
    cout << &a << endl; // print address of where pointer lives
    cout << &(a[1]) << endl; // print address of 2nd element in array
    for (int i = 0; i < NUM_ELEM; i++)
        cout << a[i] << endl; // print contents of all elements
    return 0;
}
```

Demo
Static vs. Dynamic 1-D arrays...

- Compile-time allocation on stack
- Constant, self-referential
- You cannot change where array points

Dynamic runtime:
- `int array[], array[] = new int[Enum.Players]`
How does freeing memory work?

```c
int *p, *q;
p=new int;
q=new int[5];
delete p;
delete [] q;
```
What are the similarities/differences?

- **String Object vs. C String**
  - Which library to include?
    - `<string>` **vs.** `<string.h>` or `<cstring>`
  - How do we create it?
    - `string str_obj;` **vs.** `char str_arr[20];`
  - How do we access it?
    - `str_obj.at(3)` or `str_obj[3]` **vs.** `str_arr[3]` or `*(str_arr+3)`
  - How do we get the length?
    - `str_obj.size()` or `str_obj.length()` **vs.** `strlen(str_arr)`
  - How is length of string determined?
    - `Size member variable` **vs.** `'\0'`, null character at end
```cpp
#include <iostream>
#include <cstring>  // or cstring
using namespace std;
#define NUM 5

int main() {
    // char str[NUM];  // create a static array
    char *str;  // create a dynamic array
    int num;

    while(1){
        cout << "how many chars: ";
        cin >> num;
        str = new char[num];  // on the heap
        cout << "enter a message: ";
        cin >> str;  // automatically put '\0' on end
        cout << &str << endl;  // pointer is on stack
        cout << str << endl;  // should be address, but cout gets smart
        cout << (void*) str << endl;  // don't get smart and give address
        cout << strlen(str) << endl;  // length of str based on where null is
        delete [] str;  // delete array off heap for no mem leak
        cout << str << endl;  // str contents and array not cleared on delete
    }
    return 0;
}
```
Passing a 1-D Array (Static/Dynamic)

```cpp
int main() {
    int array[5];
    ...
    pass_1darray(array);
    ...
}
void pass_1darray(int *a) {
    cout << "Array at zero: " << a[0] << endl;
}
OR
void pass_1darray(int a[]) {
    cout << "Array at zero: " << a[0] << endl;
}
```