CS 161
Intro to CS I

Functions and 1d Arrays/Intro to Multidimensional Arrays
Odds and Ends

• Work on Assignment #5
  - KISS!
• Exercise #7 posted...

Hint: Ascii chart

① A TD Attention to detail

② poor planning on your end does not constitute an emergency on my end.
Passing a 1-D Array (Static/Dynamic)

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;
}
2. What if I want to change where pointer points.

Static

int ar[4];

c fun( &ar),

void fun(int *a);

Should you be changing what ar points to? Compiler stops you from this!!!

Dynamic

int *ar = new int[4];

c fun( &ar),

void fun(int *a);

We can change where ar points using a

*ar = new int[10];

take me to ar contents! make sure you delete it=NEW

(*a)[1] = 100;

take me to ar's 2nd element

OSU Oregon State University
Creating Memory in Functions

Advantages to Dynamic Memory
int *i=Null; //created in main function

create_mem(&i); //call in main void
create_mem(int **m) {
    *m = new int[4];
}
OR
i = create_mem(); //call in main
int * create_mem() {
    return new int[4];
}
Demo...

```cpp
#include <iostream>
#include <cstdlib>

using namespace std;

#define NUM 5

// passing name of 1d array is type *, and we can change elements
void fun(int *a){
a[1]=100;
}

// passing address of array name adds * to type, we can change pointer
// or elements in array
void fun(int **a){
    // if array is pointing to something other than NULL, then delete before
    // making it point somewhere different, otherwise you have a mem leak!!!
    if(*a!=NULL)
        delete [] *a;  // delete the array that a in main points to, not local a
    *a=new int[5];  // make a in main point to new array on heap
    (*a)[0]=100;    // change contents of array element
}
```

OSU Oregon State University
```c
24 int main() {
25     int *a=NULL, ar[NUM]; //create a dynamic array and static array
26
27     for(int i=0; i<NUM; i++)
28         ar[i]=2; //initialize all elements to 2
29     for(int i=0; i<NUM; i++)
30         cout << ar[i] << endl; //make sure all elements are 2
31     fun(ar); //pass name to change elements
32     for(int i=0; i<NUM; i++)
33         cout << a[i] << endl; //make sure 2nd element is 100
34     //ar=new int[5]; //you can't change a constant pointer for static array
35     //fun(&ar); //compiler won't let you pass address of static array!!!
36
37     //create dynamic array on heap
38     a=new int[3];
39     //you can pass address of dynamic array because you can change where
40     //it points, as well as the elements
41     fun(&a);
42     for(int i=0; i<NUM; i++)
43         cout << a[i] << endl; //print to make sure we created it and changed [0]
44     return 0;
45 }
```
What About Memory Leaks?

- What happens here...

```cpp
int main () {
    int *i=NULL; //created in main function
    while(1) {
        i = create_mem(); //call in main
    }
}

int * create_mem() {
    return new int[4];
}
```
Fixing Memory Leaks...

• What happens here...

... int main () {
    int *i= NULL; // created in main function
    while(1) {
        i = create_mem(); // call in main
        delete [] i; // free memory that i points to, preventing mem leaks
    }
}
int* create_mem(){
    return new int[4];
}