CS 161
Intro to CS I
Continue Arrays
Odds and Ends...

• Last week to demo Assignment #4
• Be working on Assignment #5
```cpp
#include <iostream>
#include <cstring> //string.h

using namespace std;

int main() {
    //char str[20]; //statically allocated can only hold 19 chars entered
    //instead of limiting the number of chars entered by the user, we can
    //grow our array to hold the right amount of chars
    char s; //read a char at a time
    char *str=new char[1]; //create smallest c-string, which is just null char '\0'
    char *temp; //need to capture old string in pointer

    //add '\0' to smallest c-string with length zero, but with one element
    str[0]='\0';
    cout << "Length: " << strlen(str) << endl;

    //read one character from input and add to string if it is not the newline/enter
    cin.get(s);
    while(s!='\n'){
        temp=str; //point to old string, so we don't lose it
        str=new char[strlen(temp)+2]; //add space to the array, strlen doesn't count null char + extra
        strcpy(str,temp); //copy old string into new string (destination, source)
        str[strlen(temp)]=s; //add new character where null was in the old string
        str[strlen(temp)+1]='\0'; //add null character to the last element in array
        delete [] temp; //get rid of old array, so we don't have memory leak
        cin.get(s); //get another character to make sure it isn't '\n' to stop reading input
    }

    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;
}
```
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];
}
What About Memory Leaks?

• What happens here...
...
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];  
}
```cpp
#include <iostream>
using namespace std;

void print_array(int ele, int *a) {
    for(int i=0; i<ele; i++)
        cout << a[i] << endl;
}

void set_array(int ele, int a[])
    for(int i=0; i<ele; i++)
        a[i]=5;
}

void create_array(int &ele, int *&a) {
    cout << "how many elements: ";
    cin >> ele;
    a=new int[ele];
    cout << &a << endl;
    cout << a << endl;
}

void create_array(int *ele, int **a) {
    cout << "how many elements: ";
    cin >> *ele;
    *a=new int[*ele];
    cout << &(*a) << endl;
    cout << *a << endl;
}

int main() {
    int *a, ele;

    //create_array(ele, a); //pass ele and array pointer by reference to change pointer
    create_array(&ele, &a); //pass ele and array pointer by explicit address of to change
    //passing the name of the array allows the contents to be used or changed
    set_array(ele, a);
    print_array(ele, a);

    delete [] a;
    a=NULL; //null the pointer after you free what it points to
    cout << a << endl;
    //cout << a[0] << endl; //you shouldn't go back to a place your freed
    return 0;
```