CS 161
Intro to CS I

Continue Arrays
Odds and Ends...

• Demo Assignment 4
• Assignment 5 posted
Static vs. Dynamic 1-D arrays...

- Static array: 0, 32, 64
- Dynamic array: 0 to 256
How does freeing memory work?

int *p, *q;

p=new int;
q=new int[5];
delete p;
delete [] q;
Demo
Passing a 1-D Array (Static/Dynamic)

```c
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

```cpp
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...
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...

... 

```cpp
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];
}
```
In-class Exercise

• Get into groups of 4-5.

• Write a function that takes two arrays, x and y, stores the sum of each element from x and y in a new array, z, and returns the new array z.

  – What information does your function need?
  – What does the code inside the function look like?
  – How are you going to call this function?
Multidimensional Arrays

• data_type array_name[rows][cols];
  – int array[2][3];
  – int array[4][2][3];
  – int array[2][4][2][3];

• What are examples of these?
  – 2-D – Matrices, Spreadsheet, Minesweeper, Battleship, etc.
  – 3-D – Multiple Spreadsheets, (x, y, z) system
  – 4-D – (x, y, z, time) system
Initializing 2-D Arrays

• **Declaration:** int array[2][3] = {{0,0,0},{0,0,0}};

• **Individual elements:**
  array[0][0]=0; array[0][1]=0;
  array[0][2]=0; array[1][0]=0; array[1][1]=0; array[1][2]=0;

• **Loop:**
  
    for(i = 0; i < 2; i++)
    
      for(j = 0; j < 3; j++)
      
        array[i][j]=0;

• **Why do we need multiple brackets?**
Reading/Printing 2-D Arrays

• Reading Array Values
  
  ```
  for(i = 0; i < 2; i++)
      for(j = 0; j < 3; j++) {
          cout << “Enter a value for ” << i << “, ” << j << “: ”;
          cin >> array[i][j];
      }
  ```

• Printing Array Values
  
  ```
  for(i = 0; i < 2; i++)
      for(j = 0; j < 3; j++)
          cout << “Array: ” << array[i][j] << endl;
  ```
Demo...