DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

Lists

CS179.14A Survival Guide II

Lecture Time!

- Review: Dynamic Allocation
- Lists
- Arrays of Lists of Pointers

DISCS

Review: Dynamic Allocation

- When you need to defer initialization for any reason, you will have to rely on *dynamic allocation*
  - array size known only during run-time
  - nodes in linked lists
  - etc.

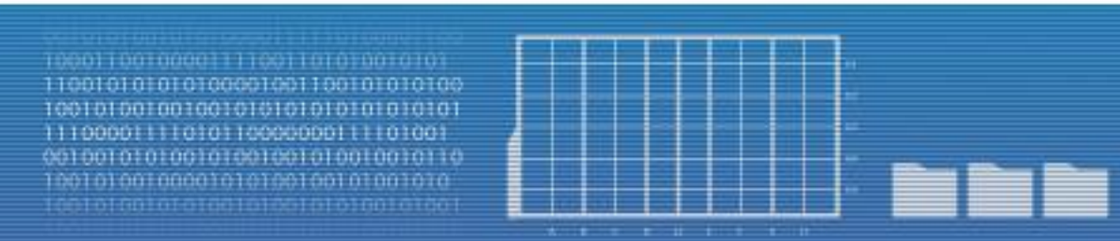
Review: Dynamic Allocation

- However, these variables are explicitly allocated through the `new` keyword
- And they must also be explicitly deallocated through the `delete` keyword
- Failure to properly deallocate will result in memory leaks
- The O/S usually deallocates them once the program terminates

Dynamic Allocation

```
int* p = new int;
*p = 28;
cout << *p << "\n";
delete p;
```

```
int size = 5;
int* arr = new int[size];
arr[1] = 12345;
cout << arr[1] << "\n";
delete[] arr;
```



Lists

- A *list* is a sequence container implemented as a doubly-linked list
- You can also use another container type (`vector`, etc.) that can act as a resizable array, but `list` generally performs better in inserting, extracting, and moving elements in any position within it

Lists

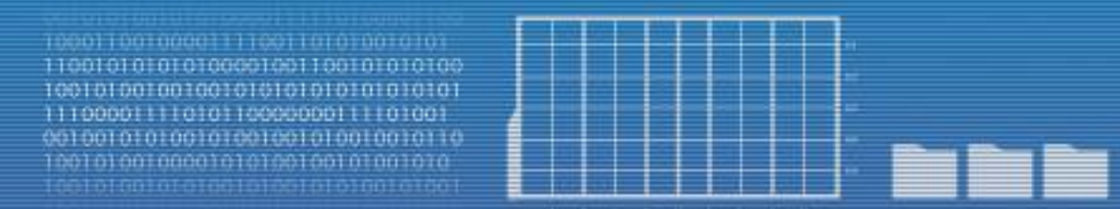
```
#include <list>

list<MyEntity*> myList;    // list of pointers

// in some function
MyEntity me;
MyEntity* p = &me;
// ...
myList.push_back( p ); // add p at end of list
myList.remove( p );  // remove all equal to p
```

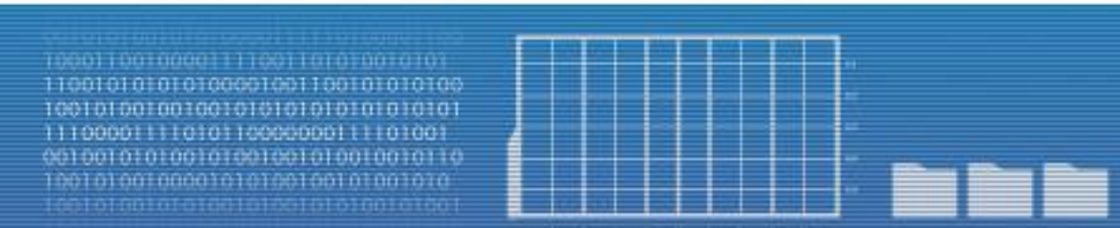
Lists

```
// ...
for(
list<MyEntity*>::iterator it = myList.begin();
it != myList.end(); ++it )
{
        cout << (**it).hp << "out of ";
        cout << (**it).maxhp << " HP left.\n";
}
```



Array of Lists

- But what if you want an array of lists?
  - one list for each node in a tree
  - one list for each cell in a grid
  - … etc.
- And what if that array's size is to be determined at run-time?

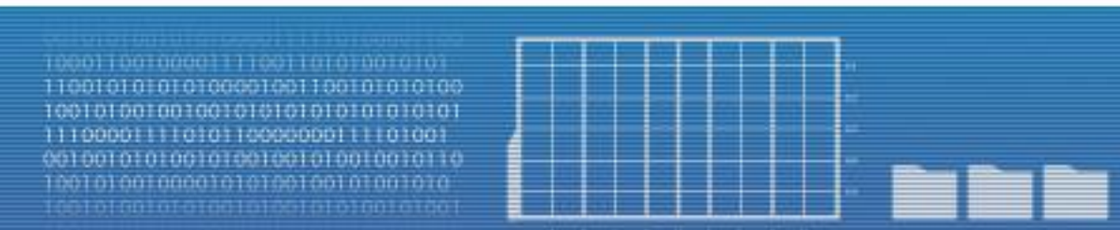


Array of Lists

```
list<MyEntity*>* row;


// in some function
row = new list<MyEntity*>[SIZE];

// for 2-D arrays, create an array of arrays
// of lists, then initialize each of these
// sub-arrays
// (there's a reason I named the variable "row")
```



Array of Lists

```
// upon program termination
delete[] row;


// for 2-D arrays, you must delete each of the
// sub-arrays first before deleting the main
// array
```