CSCI 104L Lecture 1

An Abstract Data Type (or ADT) explains **what** you want, but not **how** you achieve it. It kind of looks like a header file. It states which functions are used to interact with the data.

- A map ADT:
  1. Add(key,value)
  2. Remove(key)
  3. Lookup(key)
- Here are the key learning goals of this class:
  1. Learn techniques for how to implement data structures which are efficient. A lot of this will draw on the mathematical material from CSCI 170.
  2. Learn how to identify which data structure will best suit your needs.
  3. Learn to think about code abstractly, separating the "what" from the "how"
  4. Learn to utilize ADTs to specify the functionality of what you want.
  5. Learn good programming practice in Object-Oriented Design.

What gets output in this code? Do any lines result in errors?

```
int m = 10;
if (true) {
  int n = 3, m = 5;
  cout << n << m;
}
cout << n << m;
```

What gets output in this code?

```
int x[10];
double m = 32.33;
cout << x[10];
```

Why doesn't this code work?

```
int len;
cin >> len;
int array[len];
```

What does each line of code accomplish below? Do any result in compile errors or runtime errors?

```
int *p, *q, i = 5, j = 10;
p = i;
p = &i;
cout << p;
cout << *p;
*p = j;
*q = *p;
q = p;
```

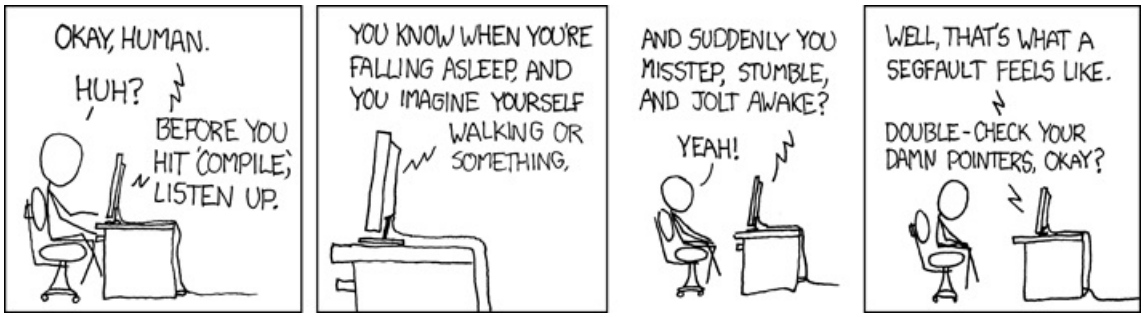

Figure 1: XKCD # 371 **Compiler Complaint**

How to allocate memory dynamically:

```
int n;
int *b = NULL;
cin >> n;
b = new int[n]; //if we only wanted one int, we could say 'b = new int;'
//...
delete [] b; //if we had b as only one int, we could say 'delete b;'
b = NULL;
```

What's wrong with the following code?

```
int i = 0;
while (true) {
  int *b;
  b = new int[100];
  //...
  b = new int[200]; //The array wasn't big enough.  Make it bigger!
  //...
  delete [] b;
  b = NULL;
  cout << i << endl;
  i++;
}
```

As a general statement: One new = one delete. This rule gets more complicated as we get to more advanced object-oriented programming.