Code Handout to accompany Thursday's final exam CSCI 455 [F13 Bono]

`String` class (selected methods)

`char charAt(int index)`
Gets the char at position index (counting from 0)

`int length()`
The number of characters in the string.

`String substring(int beginIndex, int endIndex)`
Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`.

`String substring(int beginIndex)`
Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the end of this string.

`ArrayList<ElmtType>` class (selected methods)

```
ArrayList<Integer> arrList = new ArrayList<Integer>();
                                    // create empty arraylist that can hold ints
arrList.add(3);                     // add a value to the end of the arrList
int num = arrList.get(i);           // returns element in in arrList
arrList.set(i, 7);                  // update element in arrList
int howMany = arrList.size());      // number of elements in arrList
boolean empty = arrList.isEmpty();  // true iff arrList has no elements
```

`Math` class (selected methods)

`static double floor(double a)`
Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer. e.g., Math.floor(3.2) → 3.0; Math.floor(3.0) → 3.0

`static double ceil(double a)`
Returns the largest (closest to positive infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. (short for ceiling) e.g., Math.ceil(3.2) → 4.0; Math.ceil(3.0) → 3.0

`static int min(int a, int b)`
Returns the smaller of two int values.

`static int max(int a, int b)`
Returns the greater of two int values.

C++ `Node` type and `ListType` (this is the only part of the code handout with C++ code):

```
struct Node {
  int data;
  Node * next;
  Node() { data = 0; next = NULL; }
  Node(int d) { data = d; next = NULL; }
  Node(int d, Node * n) { data = d; next = n; }
};

typedef Node * ListType;
```