

6.00 Handout, Lecture 9
(Not intended to make sense outside of lecture)

```
def bsearch(s, e, first, last):
    print first, last
    if (last - first) < 2: return s[first] == e or s[last] == e
    mid = first + (last - first)/2
    if s[mid] == e: return True
    if s[mid] > e: return bsearch(s, e, first, mid - 1)
    return bsearch(s, e, mid + 1, last)
```

```
def search1(s, e):
    print bsearch(s, e, 0, len(s) - 1)
```

```
def mergesort(list):
    if len(list) < 2: return list
    else:
        middle = len(list) / 2
        left = mergesort(list[:middle])
        right = mergesort(list[middle:])
        return merge(left, right)
```

```
def merge(left, right):
    result = []
    i, j = 0, 0
    while(i < len(left) and j < len(right)):
        if (left[i] <= right[j]):
            result.append(left[i])
            i = i + 1
        else:
            result.append(right[j])
            j = j + 1
    while (i < len(left)):
        result.append(left[i])
        i = i + 1
    while (j < len(right)):
        result.append(right[j])
        j = j + 1
    return result
```

```
def readFloat(requestMsg, errorMsg):
    while True:
        val = raw_input(requestMsg)
        try:
            val = float(val)
            return val
        except:
            print(errorMsg)
```

```
print readFloat('Enter float: ', 'Not a float.')
```

```
def readVal(valType, requestMsg, errorMsg):
    while True:
        val = raw_input(requestMsg)
        try:
            val = valType(val)
            return val
        except:
            print(errorMsg)
```

```
print readVal(int, 'Enter int: ', 'Not an int.')
```