

6.00 Handout, Lecture 11

(Not intended to make sense outside of lecture)

```
def merge(left, right):
    """Assumes left and right are sorted lists.
    Returns a new sorted list containing the same elements
    as (left + right) would contain."""
    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 mergeSort(L):
    """Returns a new sort list containing the same elements as L"""
    if len(L) < 2:
        return L[:]
    else:
        middle = int(len(L)/2)
        left = mergeSort(L[:middle])
        right = mergeSort(L[middle:])
        return merge(left, right)

def create(size):
    intSet = []
    for i in range(size):
        intSet.append(i)
    return intSet

def hashE(e, size):
    return e%size

def insert(intSet, e):
    intSet[hashE(e, len(intSet))].append(e)

def member(intSet, e):
    return e in intSet[hashE(e, len(intSet))]
```