

CS61B Lecture #2 (updated 9/3/2004)

- **Register!** Be sure to *get an account* and register on-line (part of first lab) by Friday. Otherwise, you will be dropped!
- **Lab 0** may be submitted any time before Friday midnight.
- Handouts: HW #1, Lecture #2, extras of general-information hand-out, account forms.
- Readings for Friday: 1.10-1.14 in the blue reader.

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 1

Thinking Recursively

Understand and check `isDivisible` by *tracing one level*.

E.g., `isDivisible (13, 2)`

- Call assigns `x=13, k=2`
- Body has form 'if ($k \geq x$) S_t else S_f '.
 - Since $2 < 13$, check if $13 \bmod 2 = 0$.
 - That's not true either, so reduces to
 - Rather than tracing `isDivisible(13,3)`, instead use its *comment*:
 - * Since 13 is *not* divisible by any integer in the range 3..12 (and $3 > 1$), `isDivisible(13,3)` must be *false*, and that's our answer.
 - We're done!
- Lesson: Comments aid understanding. Make them count!

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 2

Iteration

- `isDivisible` is *tail recursive*, and so creates an *iterative process*.
- Traditional "Algol family" production languages have special syntax for iteration. Four equivalent versions of `isDivisible`:

```
if (k >= x)           while (k < x) { // ! (k >= x)
    return false;      if (x % k == 0)
else if (x % k == 0)   return true;
    return true;      k = k+1;
else                  // or k += 1, or k++ (yuch).
    return isDivisible(x, k+1);}
                    return false;
```

```
int k1 = k;           for (int k1 = k; k1 < x; k1 += 1) {
while (k1 < x) {      if (x % k1 == 0)
    if (x % k1 == 0)   return true;
        return true;  }
    k1 += 1;          return false;
}
return false;
```

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 3

More Iteration: Sort an Array

Problem. Print out the command-line arguments in order:

```
% java sort the quick brown fox jumped over the lazy dog
brown dog fox jumped lazy over quick the the
```

Plan.

```
class sort {
    public static void main (String[] words) {
        sort (words, 0, words.length-1);
        print (words);
    }

    /** Sort items A[L..U], with all others unchanged. */
    static void sort (String[] A, int L, int U) { /* TOMORROW */ }

    /** Print A on one line, separated by blanks. */
    static void print (String[] A) { /* TOMORROW */ }
}
```

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 4

Selection Sort

```
/** Sort items A[L..U], with all others unchanged. */
static void sort (String[] A, int L, int U) {
    if (L < U) {
        int k = indexOfLargest (A, L, U);
        String tmp = A[k]; A[k] = A[U]; A[U] = tmp;
        sort (A, L, U-1);      // Sort items L to U-1 of A
    }
}
```

Iterative version:

```
while (L < U) {
    int k = indexOfLargest (A, L, U);
    String tmp = A[k]; A[k] = A[U]; A[U] = tmp;
    U -= 1;
}
```

And we're done! Well, OK, not quite.

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 5

Really Find Largest

```
/** Value k, I0<=k<=I1, such that V[k] is largest element among
 * V[I0], ... V[I1]. Requires I0<=I1. */
static int indexOfLargest (String[] V, int i0, int i1) {
    if (i0 >= i1)
        return i1;
    else /* if (i0 < i1) */ {
        int k = indexOfLargest (V, i0+1, i1);
        return (V[i0].compareTo (V[k]) > 0) ? i0 : k;
        // or if (V[i0].compareTo (V[k]) > 0) return i0; else return k;
    }
}
```

Iterative:

```
int i, k;
k = i1;    // Deepest iteration
for (i = i1-1; i >= i0; i -= 1)
    k = (V[i].compareTo (V[k]) > 0) ? i : k;
return k;
```

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 6

Finally, Printing

```
/** Print A on one line, separated by blanks. */
static void print (String[] A) {
    for (int i = 0; i < A.length; i += 1)
        System.out.print (A[i] + " ");
    System.out.println ();
}
```

```
/* Looking ahead: There's a brand-new syntax for the for
 * loop here (as of J2SE 5): */
for (String s : A)
    System.out.print (s + " ");
/* Use it if you like, but let's not stress over it yet! */
```

Last modified: Fri Sep 3 14:55:20 2004

CS61B: Lecture #2 7