Public-Service Announcements I

a software development club on campus that aims to
ap between academic and practical technical knowl-
ntor passionate, driven individuals and connect them
gful industry initiatives. Each semester, our Client
with industry partners to build products ranging
ck web development to machine learning. Meanwhile,
d Team focuses on learning the essentials of soft-
oment and simultaneously develops an internal tool
e or a non-profit organization.
ooking to recruit new members this semester! Appli-
ll information can be found at our website:
ebase.berkeley.edu/"

Public-Service Announcements II

nsulting is recruiting for the Fall 2018 Semester!
21 years, our group has been solving pressing busi-
ns for clients across a variety of industries. We
ll majors and backgrounds and have developed an ex-
t cultivates not only our membersâ™ skillsets but a
stic understanding of problem-solving that they carry
to whatever industry they pursue.
ple, BC was hired to completely automate terminal
on Heathrow Airport and created a framework to
senger capacity from 35M to 75M within 10 years.
ew back and forth from London for research and to
ings.
interested in applying, check out our recruitment
e (bc.berkeley.edu/bc-join.html). Feel free to come
r two infosessions on August 28 and August 30 and
on Sproul. Applications are due at 11:59pm on Fri-
31. If you have any questions, refer to our website
.edu) or email our Internal Vice President Jessica Ji
@berkeley.edu)."

Administrivia

sure you have obtained a Unix account. If you are a
nrollment student not yet on our lists, please tell a TA
n have you added to those eligible to receive an account.

t complete Lab #1, please try to do so over the weekend
are due Friday midnight). It is especially important to
central reppository.

e not to take this course after all, please tell CalCentral
t we can adjust the waiting list accordingly.

up; due next Friday at midnight. You get credit for any
but we suggest you give the problems a serious try.

2: Let's Write a Program: Prime Numbers

java Primes U to print prime numbers through U.
a Primes 101
3 5 7 11 13 17 19 23 29
37 41 43 47 53 59 61 67 71
79 83 89 97 101

prime number is an integer greater than 1 that has no
than itself other than 1.

$N/k \geq \sqrt{N}$, for $N, k > 0$.
N then N/k divides N.

ential divisors up to and including the square root.

Plan

```
rimes {
l primes up to ARGS[0] (interpreted as an
, 10 to a line. */
c void main(String[] args) {
s(Integer.parseInt(args[0]));


l primes up to and including LIMIT, 10 to
*/
ic void printPrimes(int limit) {
ery integer, x,  between 2 and LIMIT, print it if
e(x), 10 to a line. }*/


 X is prime */
ic boolean isPrime(int x) {
{ X is prime )*/;
```


Testing for Primes

```
 boolean isPrime(int x) {

lse;

sDivisible(x, 2);  // "!" means "not"


{ is divisible by any positive number >=K and < X,
1. */
 boolean isDivisible(int x, int k) {
          // a "guard"
lse;
% k == 0)  // "%" means "remainder"
ue;
(k < x && x % k != 0)
Divisible(x, k+1);
```

Iteration

is *tail recursive*, and so creates an *iterative process*.

Algol family" production languages have special syntax
. Four equivalent versions of `isDivisible`:

```
se;
 k == 0)
e;

ivisible(x, k+1);
```

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

```
x) {
 == 0)
rue;

;
```

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


Thinking Recursively

check `isDivisible(13,2)` by *tracing one level.*

```
s divisible by
>=K and < X,
 */
oolean isDivisible...

;
 == 0)


isible(x, k+1);
```

- Call assigns `x=13, k=2`
- Body has form '`if (k >= x)` S_1 `else` S_2'.
- Since $2 < 13$, we evaluate the first `else`.
- Check if $13 \bmod 2 = 0$; it's not.
- Left with `isDivisible(13,3)`.
- Rather than tracing it, instead use the *comment:*
- Since 13 is *not* divisible by any integer in the range 3..12 (and $3 > 1$), `isDivisible(13,3)` must be *false*, and we're done!
- Sounds like that last step begs the question. Why doesn't it?

nents aid understanding.
nt!

nal Task: printPrimes (Simplified)

```
primes up to and including LIMIT. */
 void printPrimes(int limit) {
```


Using Facts about Primes

sed the Useful Facts from an earlier slide. Only have
divisors up to the square root.

ent the iterative version of `isDivisible`:

```
f X is divisible by some number >=K and < X,
hat K > 1, and that X is not divisible by
ber >1 and <K. */
tic boolean isDivisible(int x, int k) {
 = (int) Math.round(Math.sqrt(x));
k1 = k; k1 <= limit; k1 += 1) {
 k1 == 0)
n true;

lse;
```

litional (blue) condition in the comment?

printPrimes (full version)

```
primes up to and including LIMIT, 10 to
/
 void printPrimes(int limit) {


 = 2; p <= limit; p += 1) {
sPrime(p)) {
ystem.out.print(p + " ");
 += 1;
 (np % 10 == 0)
  System.out.println();


10 != 0)
n.out.println();
```


Simplified printPrimes Solution

```
primes up to and including LIMIT. */
 void printPrimes(int limit) {
 = 2; p <= limit; p += 1) {
sPrime(p)) {
ystem.out.print(p + " ");


.println();
```