CS100J 4 October 2005

More on loops. Reading: Secs 7.1–7.3

Do the self-review exercises on pp. 235 and 242!!!

Quotes for the Day:

Instead of trying out computer programs on test cases until they are debugged, one should prove that they have the desired properties.
John McCarthy, 1961, A basis for a mathematical theory of computation.

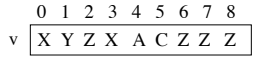
Testing may show the presence of errors, but never their absence.
Dijkstra, Second NATO Conf. on Software Engineering, 1969.

A week of hard work on a program can save you 1/2 hour of thinking.
Paul Gries, CS, University of Toronto, 2005.

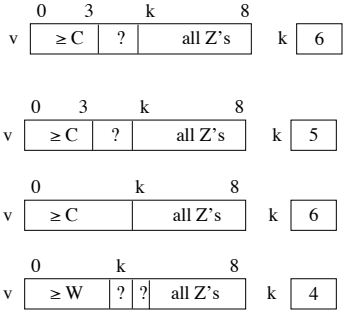
Understanding assertions

Suppose this assertion is true: **x = sum of 1..k** Under what extra condition is this one true? **x = sum of 1..n**	Put your answer here
Suppose this assertion is true: **x = sum of h..10** Under what extra condition is this one true? **x = sum of 1..10**	Put your answer here
Suppose this assertion is true: **no value in 2..k divides x** Under what extra condition is this one true? **no value in 2..n–1 divides x**	Put your answer here

Understanding assertions



This is a Vector of Characters



This is an assertion about v and k. It is **true** because chars of v[0..3] are greater than 'C' and chars of v[6..8] are 'Z's.

Indicate whether each of these 3 assertions is true or false.

The while loop

```
x= 0;
x= x + 2*2;
x= x + 3*3;
x= x + 4*4;
```

```
x= 0;
int k= 2;
while ( k != 5) {
    x= x + k*k;
    k= k+1;
}
```

To execute the while loop:

(1) Evaluate condition k != 5;

if false, stop execution of loop.

(2) Execute the repetend.

(3) Repeat again from step (1).

Repetend: the thing to be repeated. The block:

```
    {
  …
}
```


Develop loop to store in x the sum of 1..100.

We'll keep this definition of x and k true:
x = sum of 1..k–1

1. How should the loop start? Make range 1..k–1 empty: **k= 1; x= 0;**

2. When can loop stop? What condition lets us know that x has result? When **k == 101**

3. How can repetend make progress toward termination? k= k+1;

4. How do we keep def of x, h, k true? x= x + k;

Four loopy questions

```
k= 1;  x= 0;
// invariant: x = sum of 1..(k–1)
while ( k != 101) {
    x=  x + k;
    k= k + 1;
}
// { x = sum of 1..100 }
```


Develop loop to store in x the sum of 1..100.

This time, we'll keep this definition of x and k true:
x = sum of h..100

1. How should the loop start? Make range h..100 empty: **h= 101; x= 0;**

2. When can loop stop? What condition lets us know that x has result? When **h == 1**

3. How can repetend make progress toward termination? h= h – 1;

4. How do we keep def of x, h, k true? x= x + (h – 1);

Four loopy questions

```
h= 101;  x= 0;
// invariant: x = sum of h..100
while ( h != 1) {
    x=  x + (h – 1);
    h= h – 1;
}
// { x = sum of 1..100 }
```

Develop a loop (with initialization) to store in x the minimum of p*p – p for p in the range h..k.

E.g. for h..k the range –2..0, it's min of
(–2)*(–2) – 2, (–1)*(–1) – 1, 0*0 – 0

We'll keep this definition of x, h, and k true:

x = minimum of p*p – p for p in the range h..i

1. How should the loop start?
i= h; x= h*h – h;

2. When can loop stop? What condition lets us know that x has result? **i == k**

3. Make progress toward termination? i= i + 1;

4. How do we keep def of x, h, k true?
if ((i+1)*(i+1) – (i+1) < x)
x= ((i+1)*(i+1) – (i+1);

Four loopy questions

Develop a loop (with initialization) to store in x the minimum of p*p – p for p in the range h..k.

invariant: x = min of p*p - p for p in range h..i

1. How should the loop start? i= h; x= h*h – h;

2. When can loop stop? What condition lets us know that x has result? **i == k**

3. Make progress toward termination? i= i + 1;

4. How do we keep def of x, h, k true?

if ((i+1)*(i+1) – (i+1) < x)
x= ((i+1)*(i+1) – (i+1);

Four loopy questions

```
i= h;  x= h*h – h;
// invariant: x = min of p*p – p  for p
//                in the range h..i
while ( i != k) {
    if ((i+1)*(i+1) – (i+1) < x)
          x= ((i+1)*(i+1) – (i+1);
    i=  i + 1;
}
// x = min of p*p – p for p in the range h..k
```


Roach infestation!

```
/** = number of weeks it takes roaches to fill the apartment --see p 244 of text*/
public static int roaches() {
    double roachVol= .001;    // Space one roach takes
    double aptVol= 20*20*8;  // Apartment volume
    double growthRate= 1.25; // Population growth rate per week

    int w= 0;       // number of weeks
    int pop= 100; // roach population after w weeks

    // inv: pop = roach population after w weeks   AND
    //       before week w, volume of the roaches < aptVol
    while (aptVol > pop * roachVol ) {
       pop= (int) (pop * growthRate);
       w= w + 1;
    }
    return w;
  }
```


Develop loop to count the number of 'e's in String s.

We'll keep this definition of c and k true:

c = number of 'e's in s[0..h-1]

1. How should the loop start? Make range h..100 empty: **h= 0; c= 0;**

2. When can loop stop? What condition lets us know that x has result? When **h == s.length**

3. How can repetend make progress toward termination? h= h + 1;

4. How do we keep def of x, h, k true? x= x + (h – 1);

if (s.charAt[h] == 'e')
c= c + 1;

Four loopy questions

```
h= 0;  c= 0;
// invariant: c = no. of 'e's in s[0..s.length–1]
while ( h != s.length) {
    if (s.charAt[h] == 'e')   c= c + 1;
    h= h + 1;
}
// { c = no. of 'e's in s[0..h–1] }
```