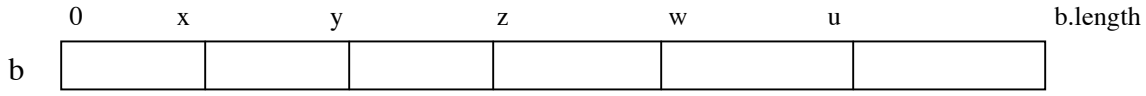
CS100J Replacement for Lecture 19. 03 November 2005

There is no lecture on 3 November. Instead, please carry out this exercise (not to be handed in), which is designed to help you learn how to develop loops from invariants. The answers are given at the end of this handout. The skill you should gain from doing this work should help you on prelim 3 and the final.

In our sample answers, we don't draw all the required pictures. If you need help, ask your TA or consultant.

1. Many of you still have problems determining a formula for the number of elements in an array segment. If it takes you more than 10 seconds to write a formula for each of the following, then you simply haven't worked on this simple task long enough. Here is one way to remember this. Below, the segment b[z..w–1] has w – z elements. Convince yourself of this by looking at the number of values when w = z, w = z+1, and w = z+2. Draw those situations. If you can remember this one cas, then you can figure out the others easily.

Write down a formula for the number of elements in each of the segments shown below:

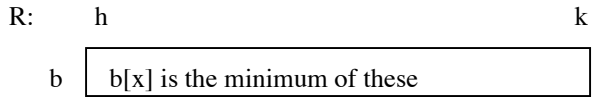


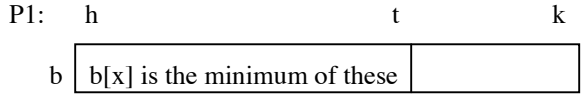
2. Draw an array that satisfies these conditions b[0..i] >= 5, b[i+1..j] = 6, b[j+1..] <= 5.

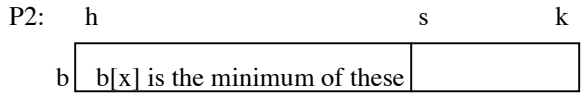
3. Below is a precondition and postcondition for the algorithm to find the min of an array b[h..k]. Below that are four different loop invariants for the problem. Write a loop (with initialization) for each one. Remember that the minimum of an empty array segment does not exist.

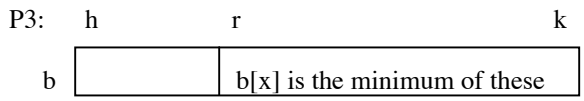
Precondition: h <= k
Postcondition R: b[x] is the minimum of b[h..k]
(a) invariant P1: b[x] is the minimum of b[h..t]
(b) invariant P2: b[x] is the minimum of b[h..s–1]
(c) invariant P3: b[x] is the minimum of b[r..k]
(d) invariant P4: b[x] is the minimum of b[w+1..k]

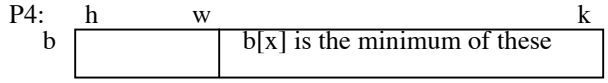
Below are the postcondition and invariants drawn as pictures. Use either the text versions or the pictures.











4. Below is a postcondition for an algorithm (which is supposed to place all the values that are ≤ 6 first) and several different loop invariants. Write a loop (with initialization) for each one. The array is not necessarily sorted. Before you begin, write the precondition (b[0..b.length–1] = ?, meaning that we know nothing about the values in the array), the postcondition, and the invariant as a picture.

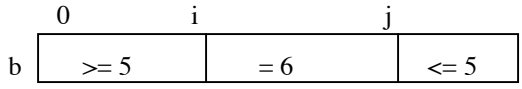
Postcondition R: b[0..k] <= 6 and b[k+1..] > 6
(a) invariant P1: b[0..h] ≤ 6 and b[k+1..] > 6
(b) invariant P2:...b[0..k] ≤ 6 and b[t..] > 6
(c) invariant P3: .. b[0..s–1] ≤ 6 and b[k+1..] > 6

5. Write selection sort, to sort array segment b[h..k], in several ways, using the invariants provided below. Before you do each one, write the invariant as a picture.

postcondition: b[0..b.length–1] is sorted (in ascending order)
(a) invariant P1: b[0..k–1] is sorted, and b[0..k–1] ≤ b[k..]
(b) invariant P2: b[0..h] is sorted, and b[0..h] ≤ b[h+1..]
(c) invariant P3: b[s+1..b.length–1] is sorted, and b[0..s] ≤ b[s+1..]

Answers to questions: 1. (x + 1) – 0, y – x, z – (y + 1), w – z, (u + 1) – w, b.length – (u + 1)

2.



3. (a)
```
t= h; x= t;
while (t != k) {
   if (b[t+1] < b[x]) x= t+1;
   t= t + 1;
}
```

(b)
```
s= h+1; x= h;
while (s-1 != k) {
   if (b[s] < b[x]) x= s;
   s= s + 1;
}
```

(c)
```
r= k; x= r;
   while (r != h) {
      if (b[r–1] < b[x]) x= r–1;
      r= r–1;
}
```

(d)
```
w= k–1; x= k;
   while (w+1 != h) {
      if (b[w] < b[x]) x= w;
      w= w–1;
    }
```

4. (a)
```
h= –1; k= b.length–1;
  // invariant: P1
 while (h != k) {
    if (b[h+1] <= 6) h= h+1;
    else { Swap b[h+1] and b[k]; k= k–1; }
 }
```

(b)
```
k= –1; t= b.length;
   // invariant: P2
   while  (t != k+1) {
      if (b[k+1] <= 6) k= k+1;
      else { Swap b[k+1] and b[t–1]; t= t–1; }
   }
```

(c)
```
s= 0; k= b.length–1;
  // invariant: P3
  while (s–1 != k) {
      if ( b[s] <= 6) s= s+1;
      else { Swap b[s] and b[k]; k= k–1; }
  }
```

5. (a)
```
k= 0;
// invariant: P1
while (k != b.length] {
   Set t to the index of the minimum
      of b[k..b.length–1];
   Swap b[k] and b[t];   k= k+1;
}
```

(b)
```
h= –1;
// invariant: P2
while (h != b.length–1]) {
   Set t to the index of the minimum
      of b[h+1..b.length–1];
   Swap b[h] and b[t];   h=  h+1;
}
```

(c)
```
s= b.length–1;
  // invariant: P3
 while (s != –1) {
    Set t to the index of the minimum of b[0..s]
    Swap b[t] and b[s];      s= s–1;
 }
```