Recitation 3:
Summary of Program Correctness.

Recitation TA Names Here

1

Ranges m..n

This notation is important because we use it to help specify, develop, and understand algorithms that manipulate arrays, strings, and lists of various forms.

Notation m..n denotes a range.
Example: b[5..7]: the segment of array b containing b[5], b[6], b[7]

b[2..3] contains 3+1-2 = 2 values, b[2], b[3]
b[2..2] contains 2+1-2 = 1 values, b[2]
b[2..1] contains 1+1-2 = 0 values. —this is important!
b[2..0] undefined! makes no sense

m..n contains n+1-m values: **Follower minus First**

2

Array notation for an assertion

b[h..k-1] ≤ 0 : an assertion that is true or false. Equivalent to
every element of b[h..k-1] is at most 0

If b[h..k-1] is (3, 0, -1, 4) the assertion is false

If b[h..k-1] is (0, 0, -1, -4) the assertion is true

If b[h..k-1] is () (i.e. h = k) the assertion is true

3

Array notation for an assertion

b[h..k-1] ≤ 0 : an assertion that is true or false. Equivalent to
every element of b[h..k-1] is at most 0

The indexes are NEVER drawn above the line
only AFTER a line or BEFORE a line

We can draw it as an array diagram:
This diagram and the assertion
b[h..k-1] ≤ 0 are equivalent!

h k
b <= 0

4

The Hoare triple

{Q} S {R} means:
Execution of S beginning with Q true will terminate in a state in which R is true.

This is false:

{x = 5} (precondition)
x= x+1;
{x = 5} (postcondition)

This is true:

{x = 5}
x= x+1;
{x > 0}

In a Java program, we might write it like this:

```
// x = 5
x= x+1;
// x > 0
```

5

The Hoare triple

{Q} S {R} means:
Execution of S beginning with Q true will terminate in a state in which R is true.

```
// x = sum of b[0..k-1]
x= x + b[k];
k= k+1;
// x = sum of b[0..k-1]
```

Here's the precondition, written as a diagram

0 k
b x = sum of these

Hey, the pre- and post-conditions are the same!
The statements keep the precondition invariantly true!

6

The assignment statement

In the videos, this was the definition of the assignment statement (in terms of its correctness):

{R[x:=e]}   x= e;   {R}

This is included only for those who are curious. *We will not require its formal use in the rest of CS2110.*

Interesting point: The definition shows how the necessary and sufficient precondition for the assignment to truthify result R can be calculated; nothing has to be guessed. Take later courses on correctness proofs and you will see this heavily used.

7

Definition of the if-statement

Suppose we want to explain when this is true: {Q} if (B) S {R}

If B is true, then execution of S has to terminate with R true. So we write: {Q and B} S {R}

If B is false, then R has to be true. So we write: Q and !B implies R

Therefore, we write:
if {Q and B} S {R} and (Q and !B implies R) then {Q} if (B) S {R}

This is just explaining in formal terms what should be common sense, based on how an if-statement is executed.

8