15-122 Bootcamp: Point-to-Proofs

Spring 2024

get materials here!

tinyurl.com/122proofsBC

Today's Agenda

get materials here!

tinyurl.com/122proofsBC

01

Quick Proof Recap

02

TA Example: Increasing A

03

Exercises: Find Min

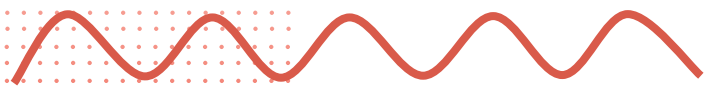
get materials here!

tinyurl.com/122proofsBC

Quick Proof Recap

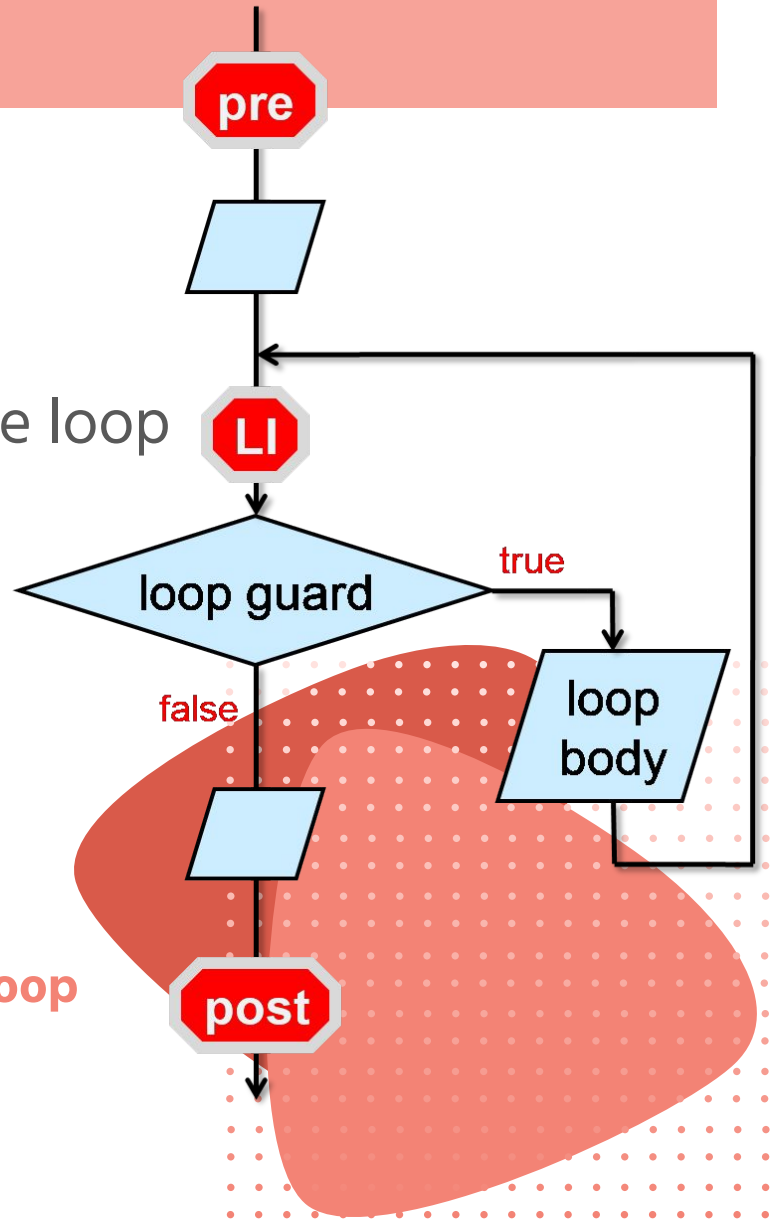
Safety (Arrays)

- We prove something is **safe** by showing that its preconditions are satisfied
  - function calls — we won't practice these today
  - primitive operations ←
  - with array accesses A[i], we need to show that the index i is in bounds
    - In other words, show that **0 <= i < \length(A)**



Correctness Proofs

- **INIT:** Proving LI is true before the loop
  - Use **code before the loop** and **preconditions**
- **PRES:** Proving the LI is true in an arbitrary iteration of the loop
  - Use **code within the loop** and the **loop guard**
- **TERM:** Proving that the loop terminates
  - This follows a template — we won't be practicing these today
- **EXIT:** proving that the loop is correct (and/or function returns correct value)
  - Use **loop invariants**, **negation of the loop guard**, **code after the loop**



TA Example:

Increasing A

Safety Example

To Show: 0 <= i - 1 && i - 1 < \length(A)

To Show: 0 <= i - 1

A.	0 <= i	by	line 7
B.	i != 0	by	short-circuiting line 8
C.	0 < i	by	math on A and B
D.	0 <= i - 1	by	math on C

To Show: i - 1 < \length(A)

E.	i <= n	by	line 7
F.	n == \length(A)	by	line 3, n unchanged
G.	i - 1 < \length(A)	by	math on **A**, E, F

```
1  // returns array containing 1, 2, 3, 4... n
2  int[] increasing_A(int[] A, int n)
3  //@requires n >= 1 && \length(A) == n;
4  //@ensures A[n-1] == n
5  {
6    for (int i = 0; i < n; i++)
7    //@loop_invariant 0 <= i && i <= n;
8    //@loop_invariant i == 0 || A[i-1] == i;
9    {
10     A[i] = i + 1;
11   }
12
13   return A;
14 }
15
```

INIT Example

To Show: i == 0 || A[i - 1] == i;

A. i = 0 by line 6

B. i == 0 || A[i - 1] == i by logic on A (short-circuiting)

```
1  // returns array containing 1, 2, 3, 4... n
2  int[] increasing_A(int[] A, int n)
3  //@requires n >= 1 && \length(A) == n;
4  //@ensures A[n-1] == n
5  {
6    for (int i = 0; i < n; i++)
7    //@loop_invariant 0 <= i && i <= n;
8    //@loop_invariant i == 0 || A[i-1] == i;
9    {
10     A[i] = i + 1;
11   }
12
13   return A;
14 }
15
```

Pres Example

Assuming: i == 0 || A[i - 1] == i
To Show: i′ == 0 || A[i′ - 1] == i′

A.	i′ = i + 1	by	line 6
B.	A[i] = i + 1	by	line 10
C.	i′ - 1 = i	by	math on A
D.	i == 0 || A[i′-1] = i′	by	math on B and C

```
1  // returns array containing 1, 2, 3, 4... n
2  int[] increasing_A(int[] A, int n)
3  //@requires n >= 1 && \length(A) == n;
4  //@ensures A[n-1] == n
5  {
6    for (int i = 0; i < n; i++)
7    //@loop_invariant 0 <= i && i <= n;
8    //@loop_invariant i == 0 || A[i-1] == i;
9    {
10     A[i] = i + 1;
11   }
12
13   return A;
14 }
15
```

Exit Example

To Show: A[n - 1] == n

A.	i >= n	by	negation of LG, line 6
B.	n == \length(A)	by	line 3, n unchanged
C.	i <= n	by	line 7, B
D.	i == n	by	A and C
E.	n != 0	by	line 3, n unchanged
F.	i == 0 || A[i - 1] == i	by	line 8
G.	A[n - 1] == n	by	logic, D, E and F

LI must be true upon exiting loop!

```
1  // returns array containing 1, 2, 3, 4... n
2  int[] increasing_A(int[] A, int n)
3  //@requires n >= 1 && \length(A) == n;
4  //@ensures A[n-1] == n
5  {
6    for (int i = 0; i < n; i++)
7    //@loop_invariant 0 <= i && i <= n;
8    //@loop_invariant i == 0 || A[i-1] == i;
9    {
10     A[i] = i + 1;
11   }
12
13   return A;
14 }
15
```

Exercises:

Find Min

Safety Example

To Show: 0 <= j && j < \length(A);

To Show: 0 <= j

A.	lo < j	by	line 8
B.	0 <= lo	by	line 2, lo unchanged
C.	0 <= j	by	math on A, B

To Show: j < \length(A)

D.	j < hi	by	line 7
E.	hi <= \length(A)	by	line 2, hi unchanged
F.	j < \length(A)	by	math on D, E

```
1  int find_min (int[] A, int lo, int hi)
2  //@requires 0 <= lo && lo < hi && hi <= \length(A);
3  //@ensures lo <= \result && \result < hi;
4  //@ensures A[\result] <= A[lo];
5  {
6     int m = lo;
7     for (int j = lo + 1; j < hi; j++)
8     //@loop_invariant lo < j && j <= hi;
9     //@loop_invariant lo <= m && m < hi;
10    //@loop_invariant A[m] <= A[lo];
11    {
12       if (A[j] < A[m])
13          m = j;
14    }
15    return m;
16 }
```

INIT Example

To Show: A[m] <= A[lo]

A.	m = lo	by	line 6
B.	A[lo] <= A[lo]	by	math
C.	A[m] <= A[lo]	by	math on A and B

```
1  int find_min (int[] A, int lo, int hi)
2  //@requires 0 <= lo && lo < hi && hi <= \length(A);
3  //@ensures lo <= \result && \result < hi;
4  //@ensures A[\result] <= A[lo];
5  {
6     int m = lo;
7     for (int j = lo + 1; j < hi; j++)
8     //@loop_invariant lo < j && j <= hi;
9     //@loop_invariant lo <= m && m < hi;
10    //@loop_invariant A[m] <= A[lo];
11    {
12       if (A[j] < A[m])
13          m = j;
14    }
15    return m;
16 }
```

PRES Example

Assuming: A[m] <= A[lo]
To Show: A[m′] <= A[lo′]

A.	A′ = A	by A unchanged
B.	lo′ = lo	by lo unchanged

Case 1: A[j] < A[m]

C.	m′ = j	by line 13
D.	A[j] < A[m] <= A[lo]	by case, assumption
E.	A[j] <= A[lo]	by math on D
F.	A[m′] <= A[lo′]	by A, B, math on C and E

Case 2: A[j] >= A[m]

G.	m′ = m	by m unchanged
H.	A[m′] <= A[lo′]	by math on B, G & assum.

```
1  int find_min (int[] A, int lo, int hi)
2  //@requires 0 <= lo && lo < hi && hi <= \length(A);
3  //@ensures lo <= \result && \result < hi;
4  //@ensures A[\result] <= A[lo];
5  {
6     int m = lo;
7     for (int j = lo + 1; j < hi; j++)
8     //@loop_invariant lo < j && j <= hi;
9     //@loop_invariant lo <= m && m < hi;
10    //@loop_invariant A[m] <= A[lo];
11    {
12       if (A[j] < A[m])
13          m = j;
14    }
15    return m;
16 }
```

EXIT Example

To Show: A[\result] <= A[lo];

A.	\result = m	by	line 15
B.	A[m] <= A[lo]	by	line 10
C.	A[\result] <= A[lo]	by	math on A and B

```
1  int find_min (int[] A, int lo, int hi)
2  //@requires 0 <= lo && lo < hi && hi <= \length(A);
3  //@ensures lo <= \result && \result < hi;
4  //@ensures A[\result] <= A[lo];
5  {
6     int m = lo;
7     for (int j = lo + 1; j < hi; j++)
8     //@loop_invariant lo < j && j <= hi;
9     //@loop_invariant lo <= m && m < hi;
10    //@loop_invariant A[m] <= A[lo];
11    {
12       if (A[j] < A[m])
13          m = j;
14    }
15    return m;
16 }
```

Thanks for Coming!

tinyurl.com/122proofsFB