Lecture 16

Cmd ::= v := AExp (v ∈ Var)

| Cmd$_1$; Cmd$_2$

| if (BExp) then Cmd$_1$ else Cmd$_2$

| while BExp do Cmd

| skip | crash!

AExp ::= ... | input

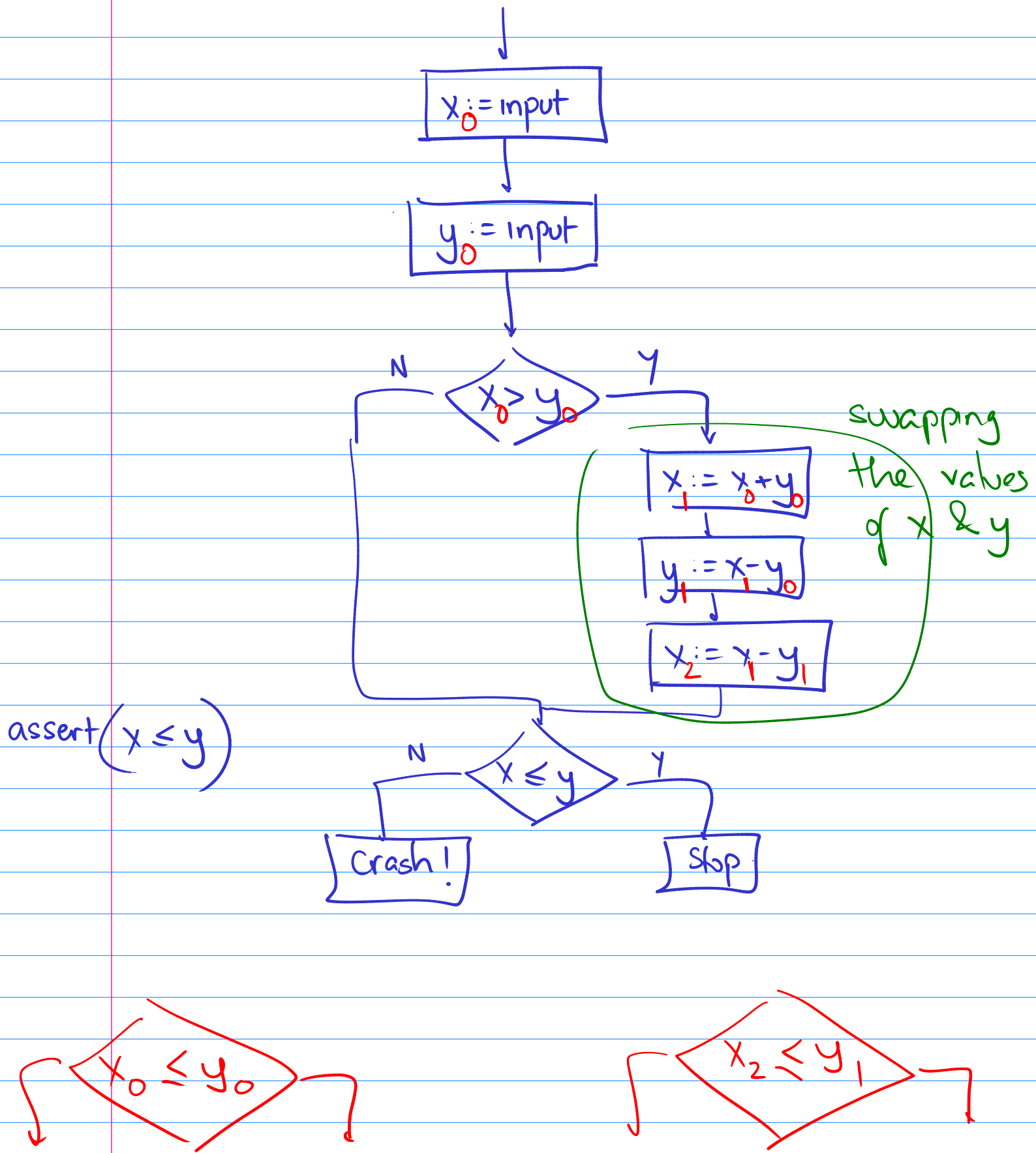
---

assert (x > 0) ⟿ if (x ≯ 0) then crash!
else skip

Outcome 1
Halt successfully

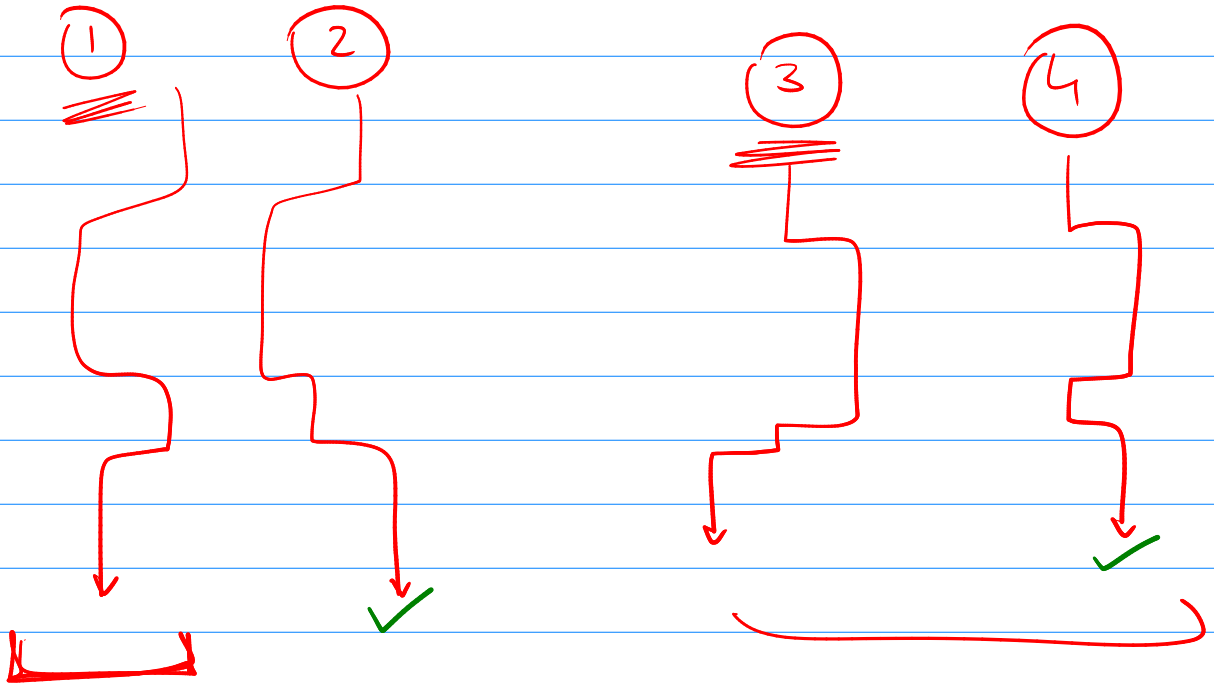
Outcome 2
Crash!

Outcome 3
Loop forever


$x_0 := input$
$y_0 := input$
$x_0 > y_0$
N
Y
$x_1 := x_0 + y_0$
$y_1 := x_1 - y_0$
$x_2 := x_1 - y_1$
swapping the values of x & y
assert $(x \leq y)$
$x \leq y$
N
Y
Crash!
Stop

$x_0 \leq y_0$

$x_2 \leq y_1$



$\exists x_0 \in \mathbb{Z} \exists y_0 \in \mathbb{Z}$

$x_0 \neq y_0$

and $x_0 \nmid y_0$

$\exists x_0 \cdot \exists y_0$

$\exists x_1 \quad \exists y_1$

$\exists x_2 \cdot (x_0 > y_0)$ and

$x_1 = x_0 + y_0$ and

$y_1 = x_1 - y_0$ and

$x_2 = x_1 - y_1$ and

$x_2 \nmid y_1$

Ex Linear search of an array

```
arr = input-array()
e = input()
i = 0 , found = 0
while (i < |arr|)
    if (arr[i] = e)
        found = 1
    i := i+1
assert( found = 1 iff ∃ j  0≤j<|arr| &
                          arr[j] = e)
```

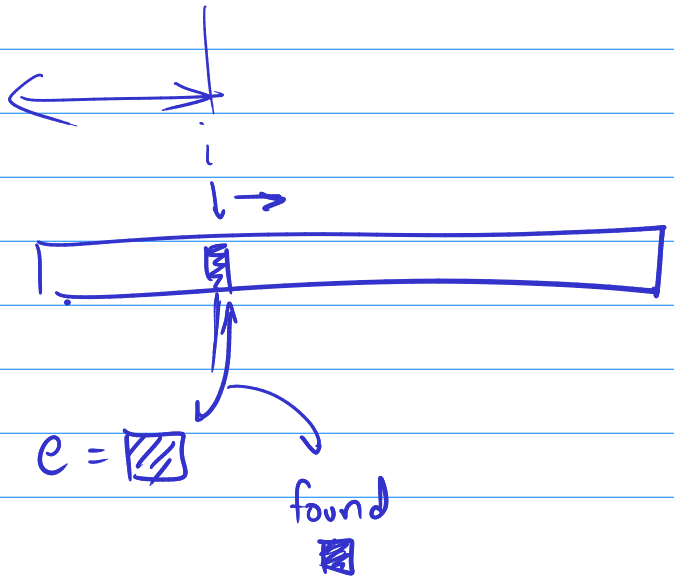
If we are able to establish this invariant, then the problem is solved

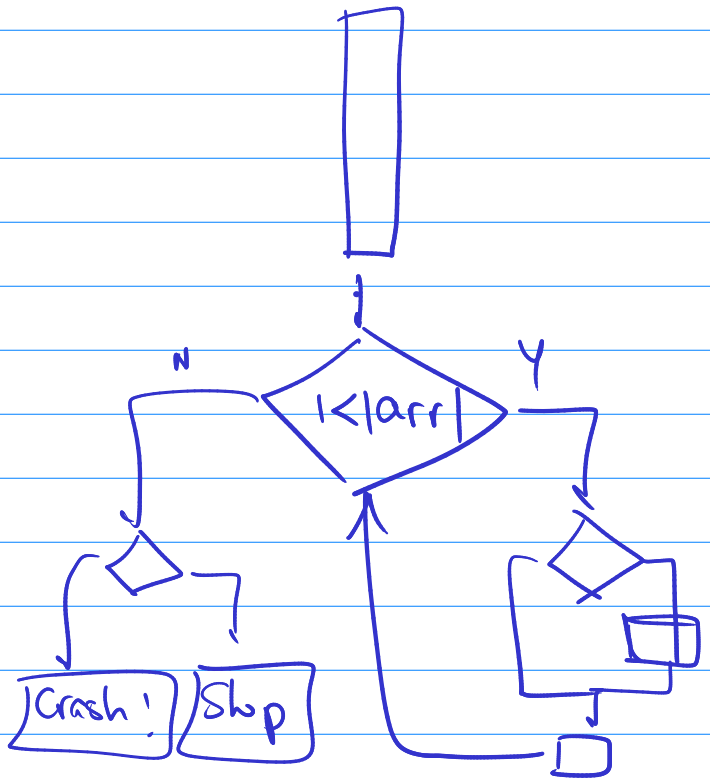
$i \leq |arr|$

found = 1 iff

$\exists j \;\; 0 \leq j < i$ &

$arr[j] = e$





- Case split based on number of times we pass through the loop

At the heart of symbolic execution

Loop unrolling / Bounded model checking

One more round of unrolling

```
while (b)        if (b) {
  e                e
                   while (b)
                     e
                 }
```

Skip unrolling

assume(!b)

Question 1: Does the program crash when |arr|=0?

Question 2: |arr|=1?

Q 3. |arr|=2?

⋮

Question 1: What do invariants "mean"?

What is the connection between invariants & induction?

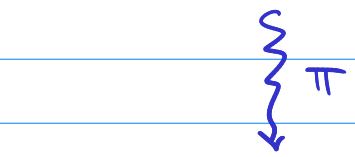
Question 2: Who comes up with invariants?

Problem: Assuming the program eventually halts, does it ever crash?

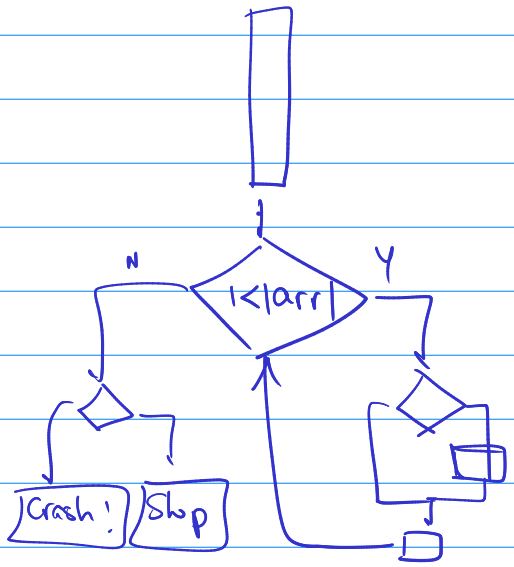
"Partial correctness".

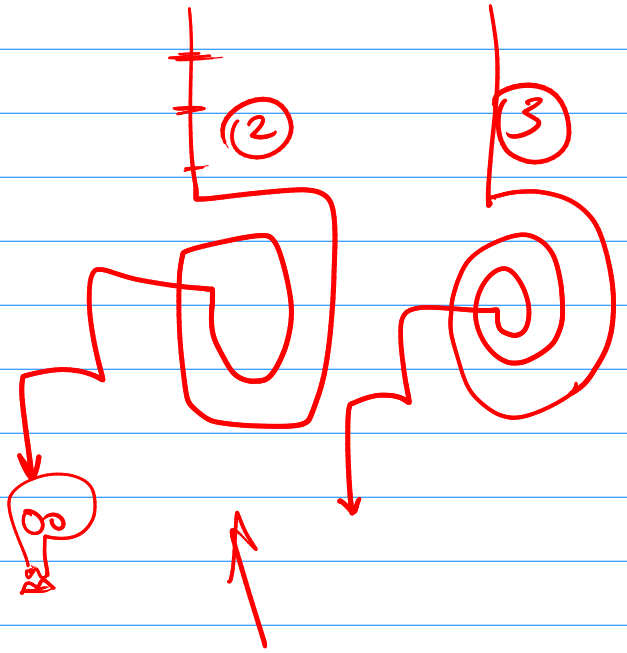
Show that

Problem: $\forall$ program paths π: Start

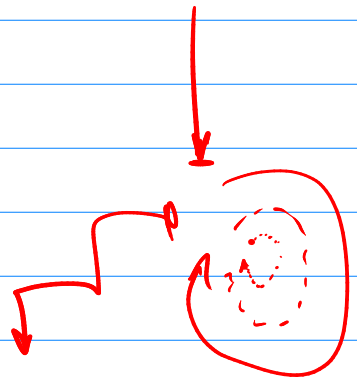


π is infeasible.

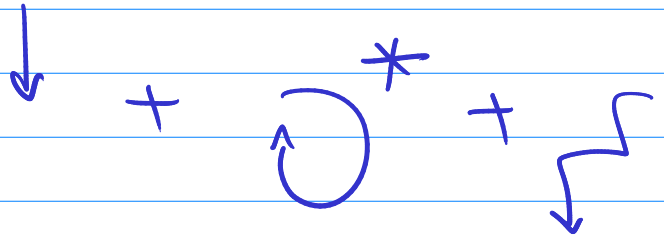




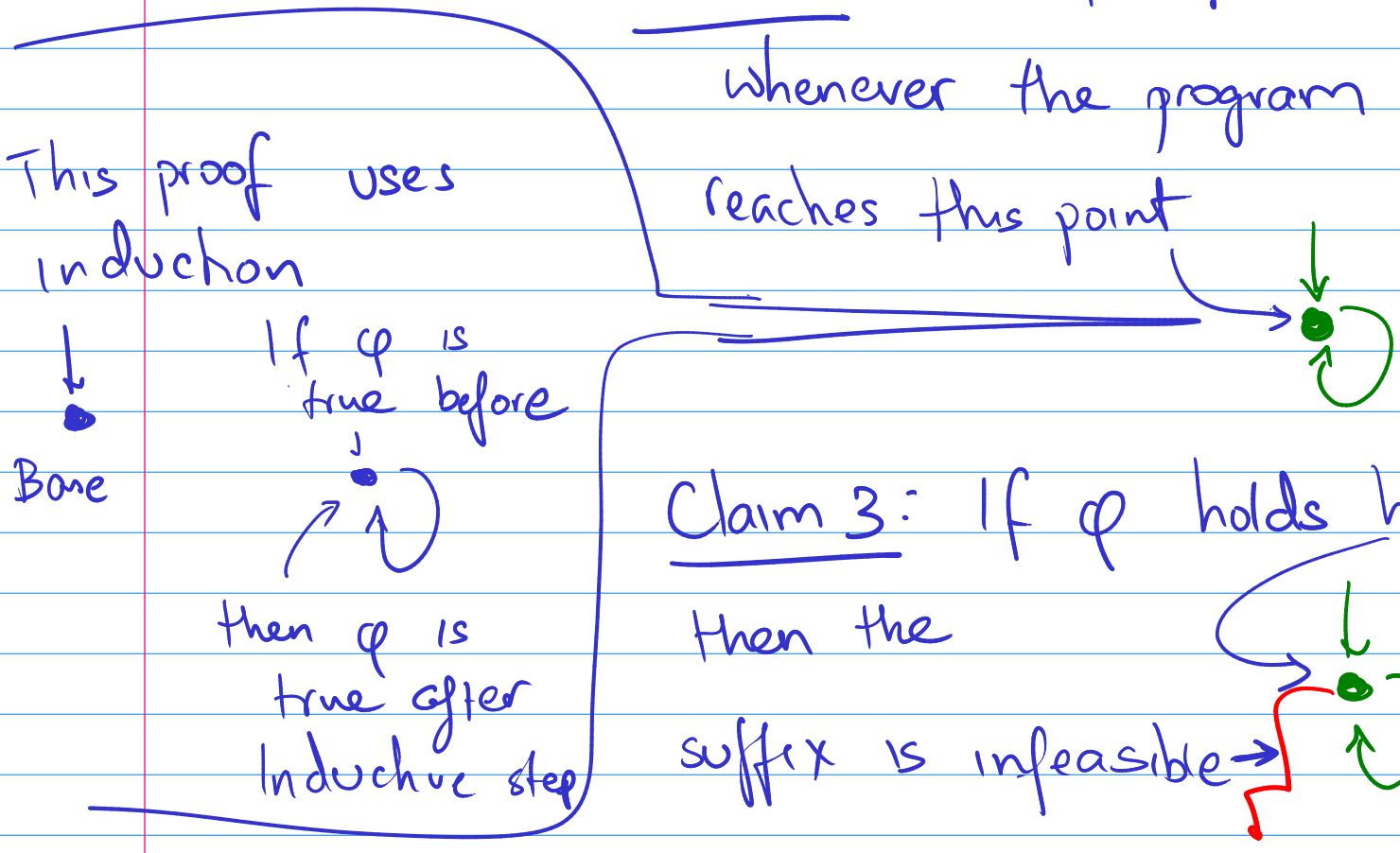
This particular flow of control abstracts away some part of the program behavior.



Claim 1: All paths can be constructed by tiling instances of the loop body



Claim 2: Some prop. φ holds whenever the program reaches this point



Claim 3: If φ holds here then the suffix is infeasible

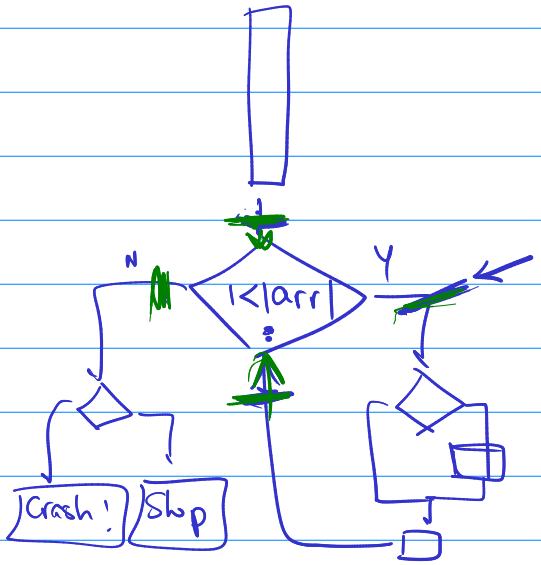
```
arr = input-array()
e = input()
i = 0 , found = 0
while (i < |arr|)
    if (arr[i] = e)
        found = 1
    i := i+1
assert( found = 1 iff ∃ j  0≤j<|arr| &
                        arr[j] = e)
```

If we are able to establish this invariant then the problem is solved

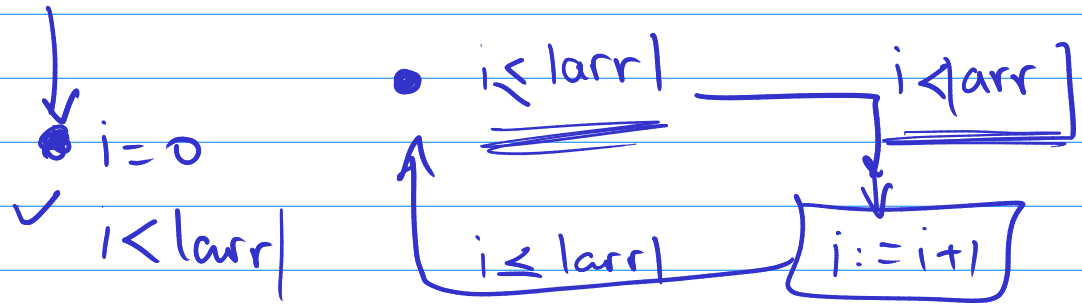
$i \leq |arr|$

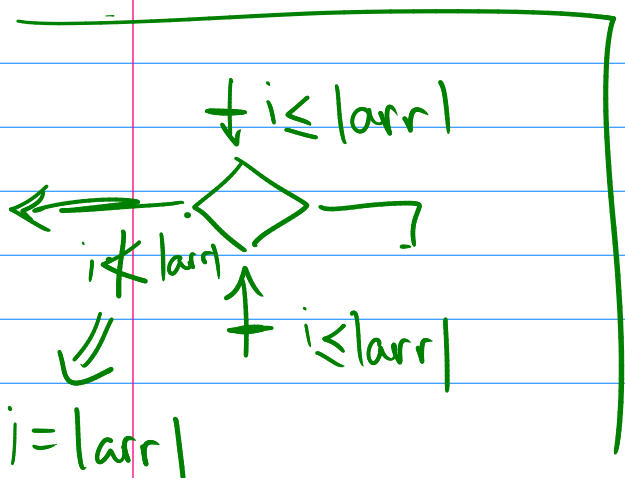
found = 1 iff

$\exists j \;\; 0 \leq j < i \;\&\; arr[j] = e$



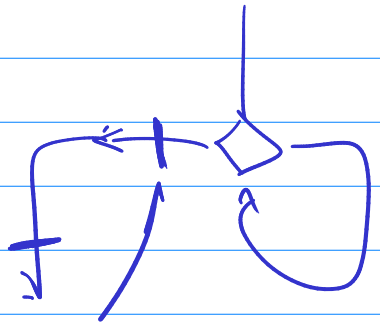
Problem: Show that $i \leq |arr|$, always.





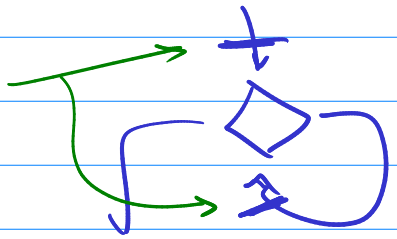
So, if the invariant holds & we go through the loop body once, then the invariant is restored

Problem



Show that $i = |arr|$ here, always.

Problem: Assume $i \leq |arr|$ here



Show that found = 1 iff

$\exists j \ 0 \leq j < i$ &

$arr[j] = e$

at the same points.

```
arr = input-array()
e = input()
i = 0 , found = 0
while (i < |arr|)
    if (arr[i] = e)
        found = 1
    i := i+1
assert( found = 1 iff ∃ j 0≤j<|arr| &
                      arr[j] = e)
```

Proof: By induction on the path.

Base case ↓

found = 0

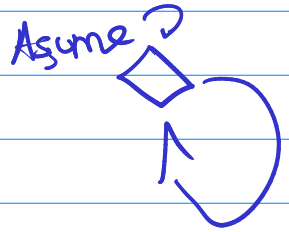
i = 0

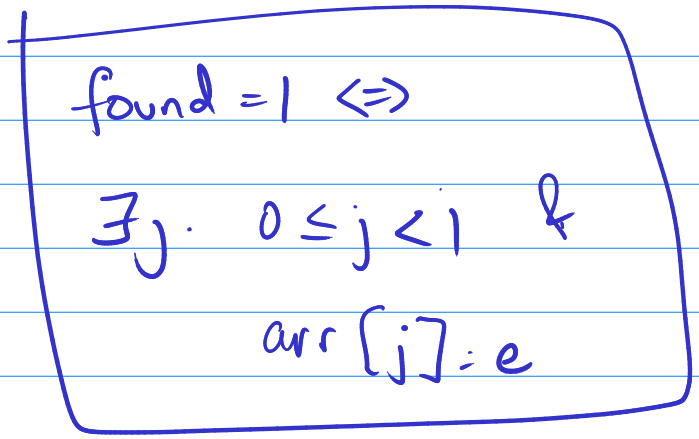
$\exists j \ 0 \leq j < i$ and $arr[j] = e$

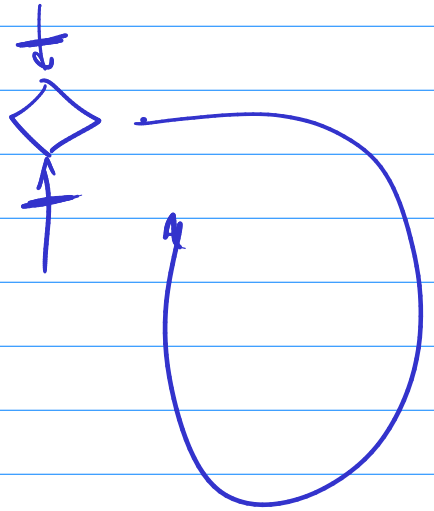
We wanted to show

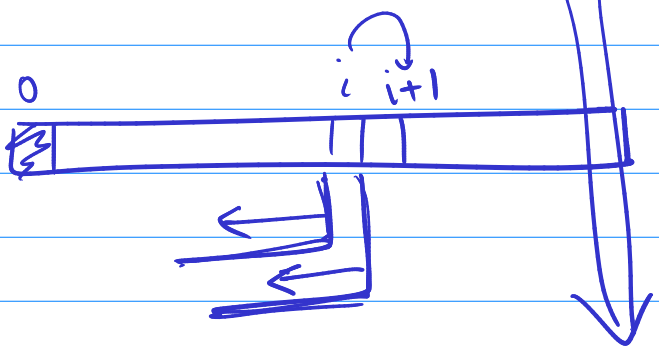
$$\underbrace{found = 1}_{false} \iff \underbrace{\exists j \;\; 0 \le j < l \;\&\; arr[j] = e}_{false}$$

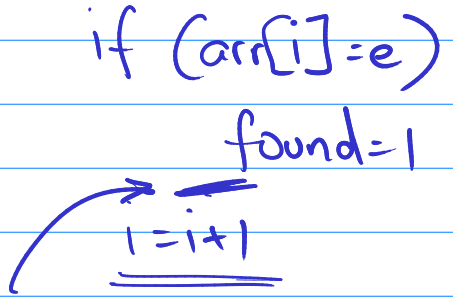
Inductive step











found = 1 ⇔
$\exists j \;\; 0 \le j \le i \;\&\; arr[j] = e$

We have established that the property is an "inductive invariant".

On Tuesday :- Floyd-Hoare logic

How to do induction on flowcharts

Developed prog calculus for Imp.

- Using program synthesis for invariant discovery

- How to rule out outcome 3: Proving that programs always halt