asm binary-search

Signature:

static	n	$: \mathbb{N}$	
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
  if a(m) = x
  then pos := m
  else if a(m) < x
  then l := m+1
  else r := m-1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

asm binary-search

Signature:

static	n	$: \mathbb{N}$	
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if 0 ≤ 3 ∧ -1 = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
let m = 1 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
    if a(m) = x
    then pos := m
    else if a(m) < x
    then l := m+1
    else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := m
else if a(1) < x
then l := 2
else r := 0
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

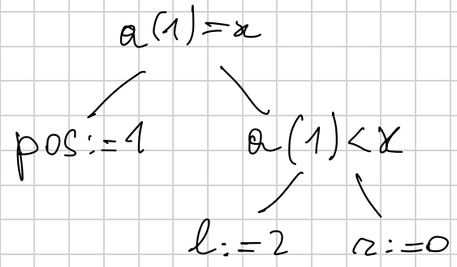
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```

seq

[Main]

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

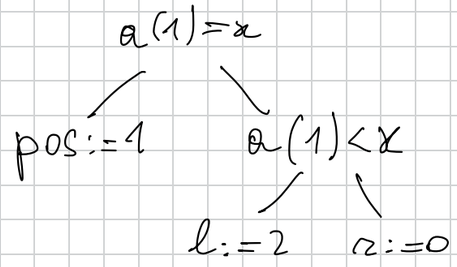
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq [Main]
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

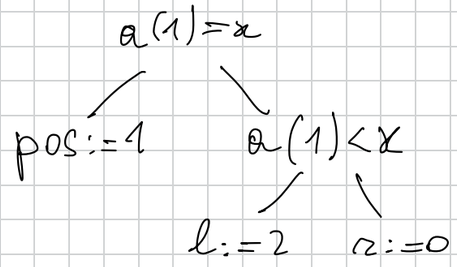
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq [Main]
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

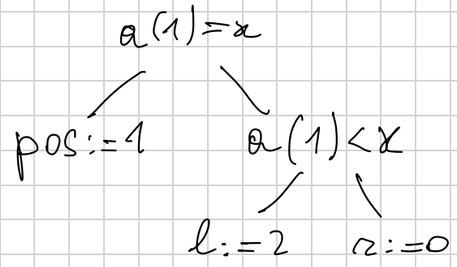
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq [Main]
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

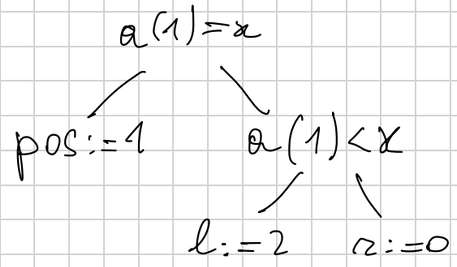
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq
        if l ≤ r ∧ pos = -1
        then let m = (l+r) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
                else r := m-1
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

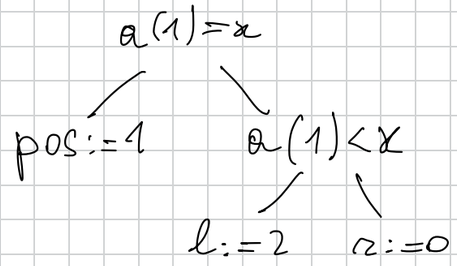
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq                    =false
        if l ≤ r ∧ pos = -1
        then let m = (l+r) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
                else r := m-1
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

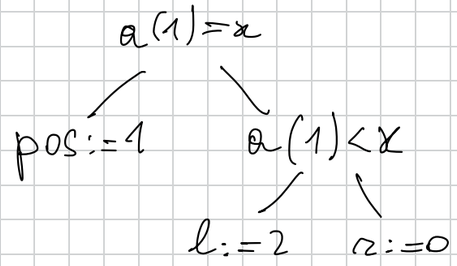
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

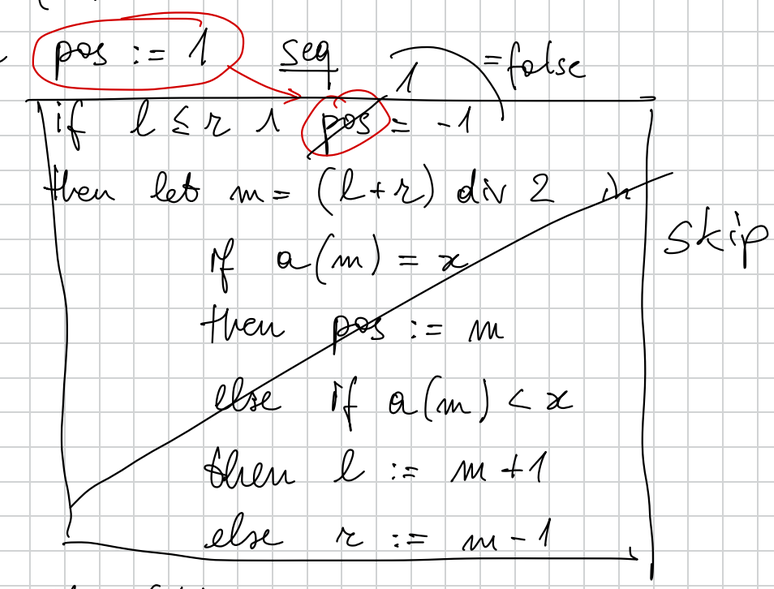
Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq if l ≤ r ∧ pos = -1
                  then let m = (l+r) div 2 in
                          if a(m) = x
                          then pos := m
                          else if a(m) < x
                          then l := m+1
                          else r := m-1
                  else skip
else if a(1) < x
then l := 2 seq [Main]
else r := 0 seq [Main]
```



asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

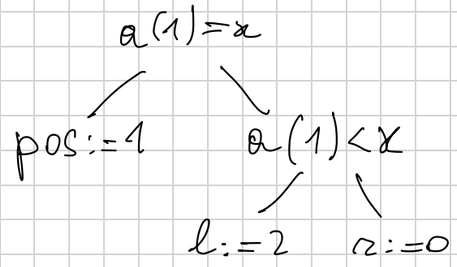
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1   seq Skip
else if a(1) < x
then l := 2   seq [Main]
else r := 0   seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

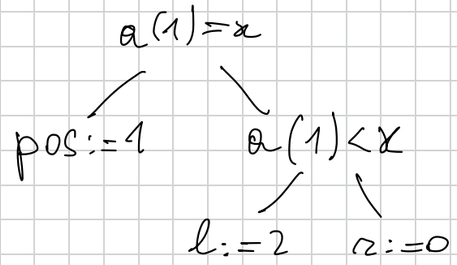
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1 seq skip
else if a(1) < x
then l := 2 seq
        if l ≤ r ∧ pos = -1
        then let m = (l+r) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
                else r := m-1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

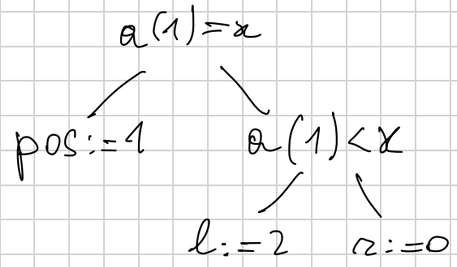
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
     if a(m) = x
     then pos := m
     else if a(m) < x
     then l := m+1
     else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
     if l ≤ r ∧ pos = -1
     then let m = (l+r) div 2 in
          if a(m) = x
          then pos := m
          else if a(m) < x
          then l := m+1
          else r := m-1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

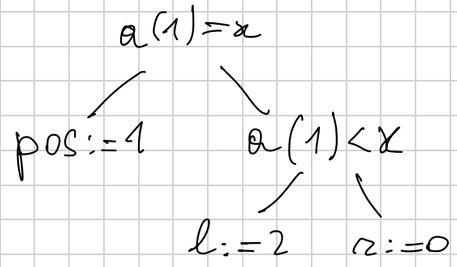
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
        if 2 ≤ 3 ∧ -1 = -1
        then let m = (2+3) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
                else r := m-1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

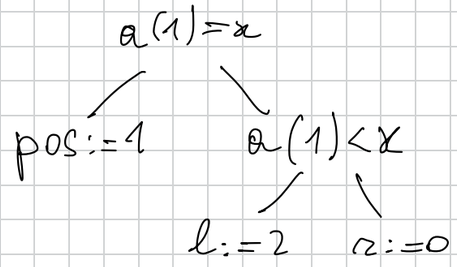
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
                          = 2
        let m = (2+3) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
                else r := m-1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

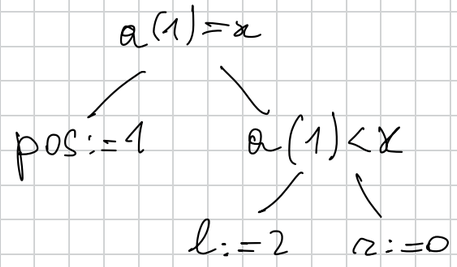
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
        if a(2) = x
        then pos := 2
        else if a(2) < x
        then l := 2 +1
        else r := 2 -1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

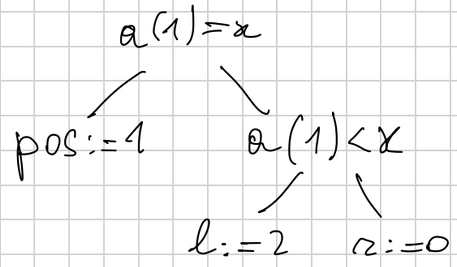
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
```

```
                if a(2) = x
                then pos := 2
                else if a(2) < x
                then l := 3
                else r := 1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

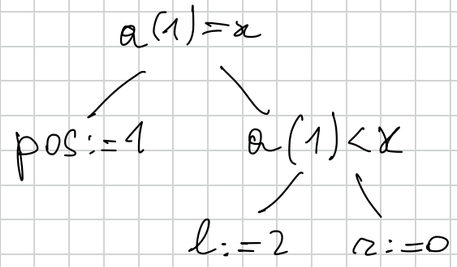
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2 seq
        if a(2) = x
        then l := 2 seq pos := 2
        else if a(2) < x
        then l := 2 seq l := 3
        else l := 2 seq r := 1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

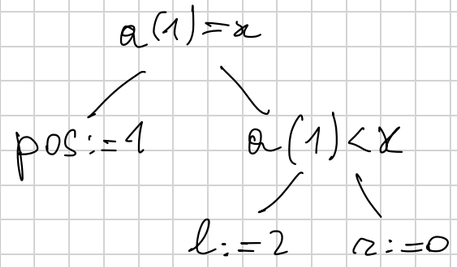
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then if a(2) = x
        then l:=2 seq pos := 2
        else if a(2) < x
        then ~~l:=2 seq~~ l := 3
        else l:=2 seq r := 1
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

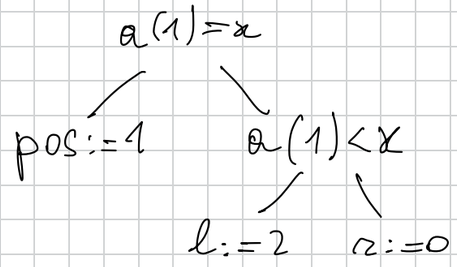
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then if a(2) = x
        then { l:=2 , pos := 2 }
        else if a(2) < x
        then l := 3
        else { l:=2 , r := 1 }
else r := 0 seq [Main]
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

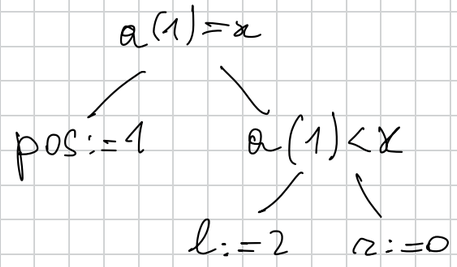
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then if a(2) = x
        then { l:=2 , pos := 2 }
        else if a(2) < x
        then l := 3
        else { l:=2 , r := 1 }
else r := 0 seq
        if l ≤ r ∧ pos = -1
        then let m = (l+r) div 2 in
                if a(m) = x
                then pos := m
                else if a(m) < x
                then l := m+1
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

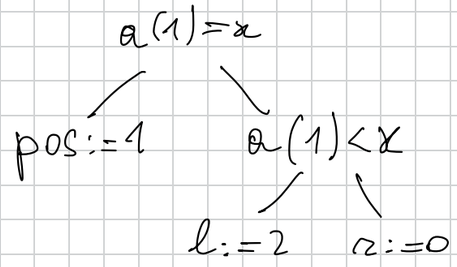
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then if a(2) = x
        then { l:=2 , pos := 2 }
        else if a(2) < x
        then l := 3
        else { l:=2 , r := 1 }
else if a(0) = x
        then { pos :=0 , r:=0 }
        else if a(0) < x
        then { l:=1 , r:=0 }
        else { r:=-1 }
```

asm binary-search

Signature:

static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

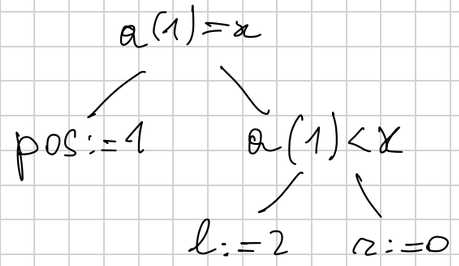
rule Main:

```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
        if a(m) = x
        then pos := m
        else if a(m) < x
        then l := m+1
        else r := m-1
```

SYMBOLIC EXECUTION

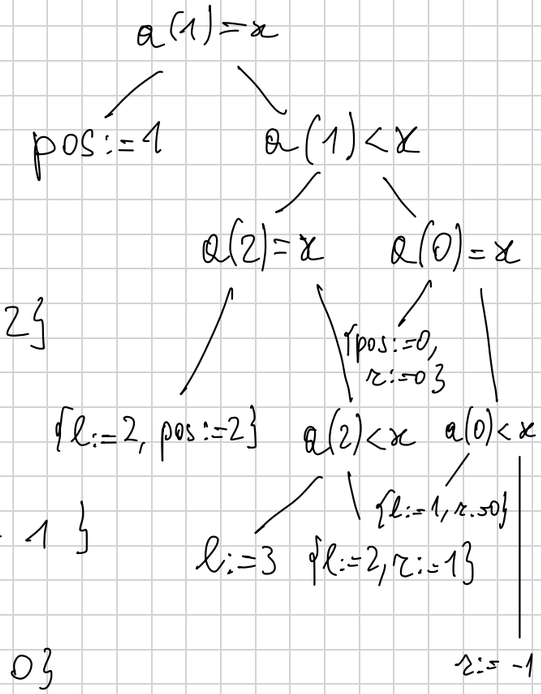
Main¹ (evaluated in S_0)

```
if a(1) = x
then pos := 1
else if a(1) < x
then l := 2
else r := 0
```



Main² = Main¹ (S_0) seq Main

```
if a(1) = x
then pos := 1
else if a(1) < x
then if a(2) = x
     then {l:=2, pos:=2}
     else if a(2) < x
     then l := 3
     else {l:=2, r := 1}
else if a(0) = x
     then {pos :=0, r:=0}
     else if a(0) < x
     then {l:=1, r:=0}
     else {r:=-1}
```



asm binary-search

Signature:

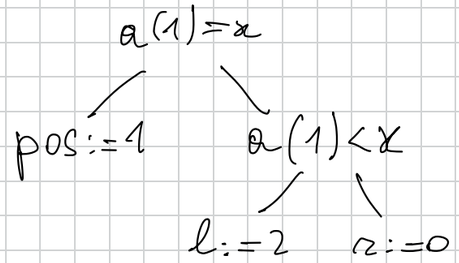
static	n	$: \mathbb{N}$	(defined as 4)
controlled	a	$: \mathbb{N} \to \mathbb{Z}$	(uninitialised)
controlled	x	$: \mathbb{Z}$	(uninitialised)
controlled	l	$: \mathbb{N}$	(initially 0)
controlled	r	$: \mathbb{N}$	(initially $n-1$)
controlled	pos	$: \mathbb{N}$	(initially -1)

rule Main:

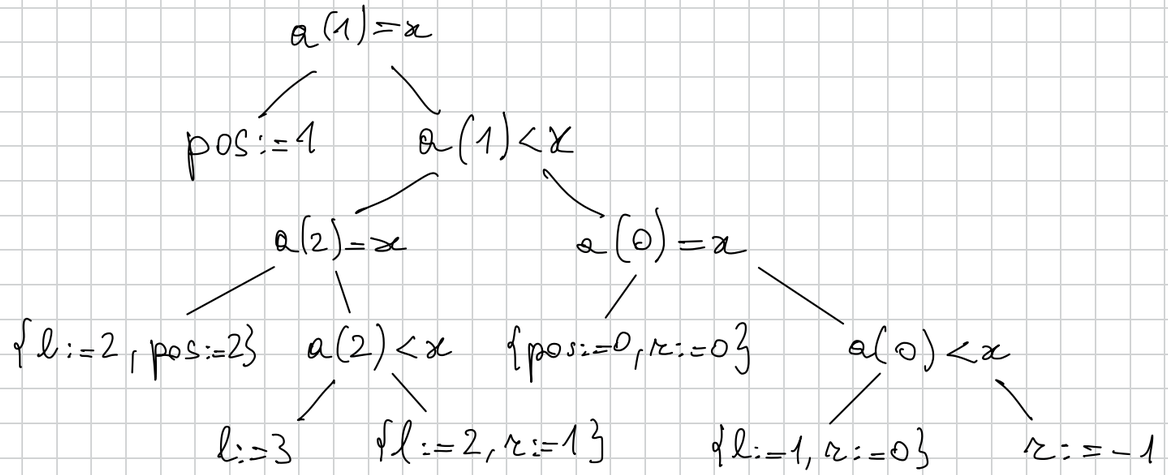
```
if l ≤ r ∧ pos = -1
then let m = (l+r) div 2 in
    if a(m) = x
    then pos := m
    else if a(m) < x
    then l := m+1
    else r := m-1
```

SYMBOLIC EXECUTION

Main¹ (evaluated in S_0)



Main² = Main¹ (S_0) seq Main



Main³ = Main² (S_0) seq Main