```
from z3 import *

def binarySearch(a: list, key: int):
    l = 0
    r = len(a) - 1
    while l < r:
        mid = (l + r) / 2
        if key < a[mid]:
            r = mid
        elif a[mid] < key:
            l = mid # spaeter hier l = mid + 1
        else:
            return mid
    return None

def checkTerm():
    l, r, mid, l1, r1 = Ints('l r mid l1 r1')
    preconditions = And(l >= 0, l < r)
    postcon1 = And(r1 < r, r1 - l < r - l)
    postcon2 = And(l1 > l,  r - l1 < r - l)

    code_logic = [mid == (l + r) / 2,
                  Implies(True,
                          And(r1 == mid, postcon1)),
                  Implies(True,
                          And(l1 == mid, postcon2))]
                          # hier spaeter l1 == mid + 1
    prove(Implies(preconditions,
                  Exists([mid, l1, r1], And(code_logic))))

checkTerm()
```

Wir haben bei den if-Bedingungen True genutzt, da die eigentlichen if-Bedingungen keinen Einfluss auf die Terminierung haben (also unabhängig davon wie das eigentliche Array bzw. die Liste aussieht, müssen ja in beiden Fällen beide Pfade terminieren und dementsprechend haben wir es vereinfacht, um es verständlicher zu machen. Die Terminierung basiert ja darauf, dass ein Intervall echt kleiner wird und das Intervall existiert unabhängig von den Listeneinträgen .