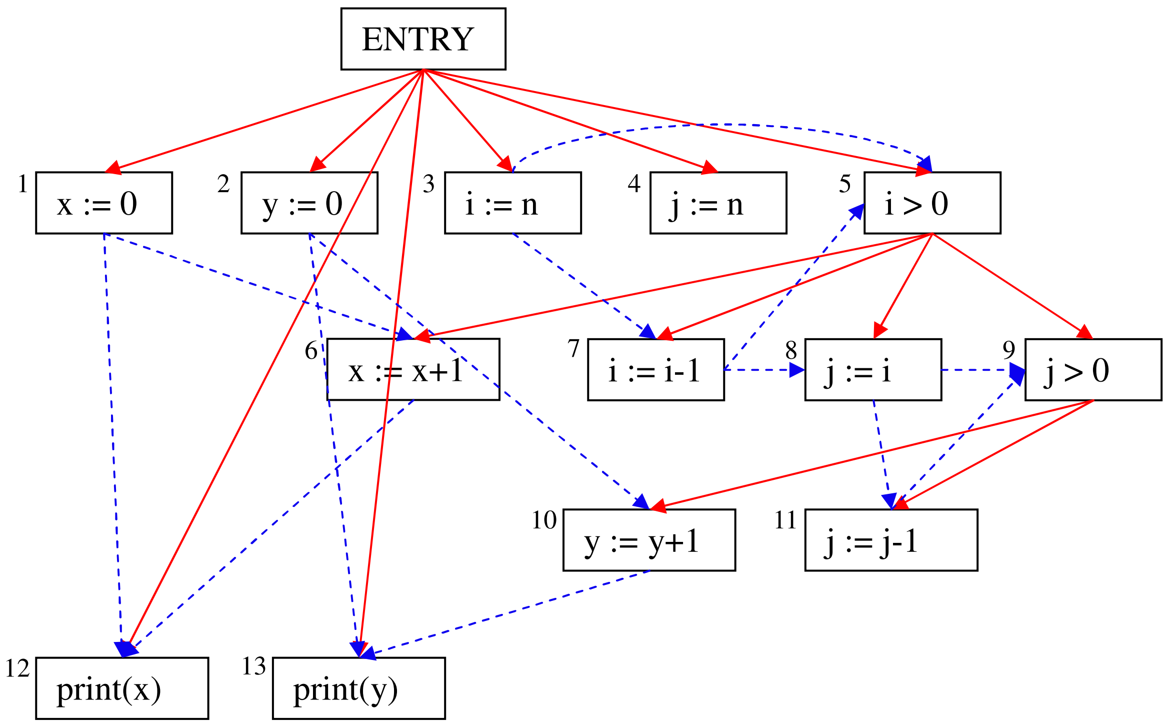
# Software Verification
## Exercise Solution: Slicing and Abstract Interpretation

## 1 Program slicing

(a)




(b)

Slicing criterion 12, i.e. print(x): ENTRY, 1, 3, 5, 6, 7, 12
i.e.
x := 0
i := n
**while** i > 0 **do**
    x := x + 1
    i := i – 1
**end**
print(x)

Slicing criterion 13, i.e. print(y): ENTRY, 2, 3, 5, 7, 8, 9, 10, 11, 13
i.e.

```
y := 0
i := n
while i > 0 do
        i := i – 1
        j := i
        while j > 0 do
                y := y + 1
                j := j – 1
        end
end
print(y)
```

Note that a slice shows which parts of the program contribute to the values of the variables that the slicing criterion statement uses (reads). This is the case because we use definition-use information to indicate data dependencies in the program dependence graph.

## 2 Abstract interpretation

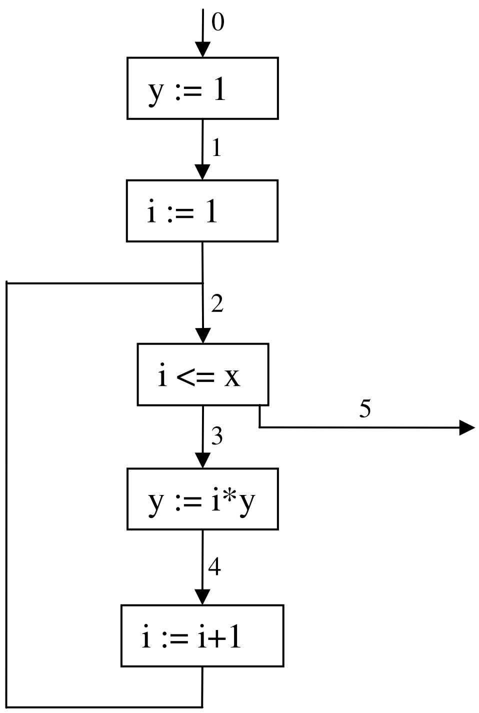
(a)

| | | Iterations | | | | | | | | | | | | | Final answer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $A_1$ | x | + | | | | | | | | | | | | | + |
| | y | $\top$ | | | | | | | | | | | | | $\top$ |
| $A_2$ | x | $\bot$ | + | | | | $\top$ | | | | $\top$ | | | | $\top$ |
| | y | $\bot$ | + | | | | + | | | | $\top$ | | | | $\top$ |
| $A_3$ | x | $\bot$ | | + | | | | $\top$ | | | | $\top$ | | | $\top$ |
| | y | $\bot$ | | + | | | | + | | | | $\top$ | | | $\top$ |
| $A_4$ | x | $\bot$ | | | + | | | | $\top$ | | | | $\top$ | | $\top$ |
| | y | $\bot$ | | | + | | | | $\top$ | | | | $\top$ | | $\top$ |
| $A_5$ | x | $\bot$ | | | | $\bot$ | | | | 0 | | | | 0 | 0 |
| | y | $\bot$ | | | | + | | | | + | | | | $\top$ | $\top$ |

Note that this analysis is not very precise - it cannot prove that y is positive when the algorithm completes (i.e. at $A_5$). The next questions fix this.

(b) 1.

Once we eliminate the problematic minus operator, the analysis becomes more precise:




$A_0 = [x \mapsto +, y \mapsto \top, i \mapsto \top]$
$A_1 = A_0[y \mapsto +]$
$A_2 = A_1[i \mapsto +] \sqcup A_4[i \mapsto A_4(i) \oplus +]$
$A_3 = A_2$
$A_4 = A_3[y \mapsto A_3(i) \otimes A_3(y)]$
$A_5 = A_2$

| | | |
|---|---|---|
| $A_0$ | x | + |
| | y | $\top$ |
| | i | $\top$ |
| $A_1$ | x | + |
| | y | + |
| | i | $\top$ |
| $A_2$ | x | + |
| | y | + |
| | i | + |
| $A_3$ | x | + |
| | y | + |
| | i | + |
| $A_4$ | x | + |
| | y | + |
| | i | + |
| $A_5$ | x | + |
| | y | + |
| | i | + |

(b) 2.

We use the domain $\wp(\{-,0,+\} \times \{-,0,+\})$ to represent the program state (x,y). This is a so-called *relational analysis*. The relational analysis is more precise because the domain can express dependencies, or relationships, between x and y.

$A_1 = \{(+,-), (+,0), (+,+)\}$
$A_2 = \{(x,+) \mid (x,y) \in A_1\} \cup \{(x,y') \mid (x',y') \in A_4 \text{ and } x \in x' \ominus +\}$
$A_3 = A_2 \cap \{(x,y) \mid x \in \{-,+\} \text{ and } y \in \{-,0,+\}\}$
$A_4 = \{(x',y) \mid (x',y') \in A_3 \text{ and } y \in x' \otimes y'\}$
$A_5 = A_2 \cap \{(0,y) \mid y \in \{-,0,+\}\}$

| | Iterations | | | | | | | Answer |
|---|---|---|---|---|---|---|---|---|
| $A_1$ | {(+,-), (+,0), (+,+)} | | | | | | ... | {(+,-),(+,0), (+,+)} |
| $A_2$ | Ø | {(+,+)} | | | {(+,+),(0,+), (-,+)} | | ... | {(+,+),(-,+), (0,+),(-,-)} |
| $A_3$ | Ø | | {(+,+)} | | | {(+,+), (-,+)} | ... | {(+,+),(-,+), (-,-)} |
| $A_4$ | Ø | | | {(+,+)} | | | ... | {(+,+),(-,-), (-,+)} |
| $A_5$ | Ø | | | | | | ... | {(0,+)} |