# Functional Programming and Verification

Prof. Dr. H. Seidl, N. Hartmann, R. Vogler WS 2018/19

**Exercise Sheet 1** Deadline: 28.10.2018

## General Information

Detailed information about the lecture, tutorials and homework assignments can be found on the lecture website[1]. Solutions have to be submitted to Moodle[2]. Make sure your uploaded documents are readable. Blurred images will be rejected. Use Piazza[3] to ask questions and discuss with your fellow students.

## Assignment 1.1 (L) Recap: Implications

Remember that an implication $A \implies B$ states an *if-then* relation: If $A$ is satisfied, then $B$ is satisfied as well. Let $x, y \in \mathbb{Z}$ and let $A, B$ be arbitrary formula. Decide, which of the following implications hold:

1. ☒ $x = 1 \implies 0 < x$
2. ☐ $x < 6 \implies x = 3$
3. ☒ $x > 0 \implies x \geq 0$
4. ☒ $x = -2 \implies x < -1 \lor x > 1$
5. ☒ $x = 0 \lor x = 7 \implies 4 \neq x$
6. ☐ $x = 1 \implies x \leq 3 \land y > 0$
7. ☒ $x < 8 \land y = x \implies y \neq 12$
8. ☐ $x = 1 \lor y = 1 \implies x > 0$
9. ☐ $x \neq 5 \implies false$
10. ☐ $true \implies x \neq y$
11. ☒ $false \implies x = 1$
12. ☐ $x \geq 1 \implies 2x + 3 = 5$
13. ☒ $A \land x = y \implies A$
14. ☒ $B \implies A \lor B$
15. ☒ $A \implies (B \implies A)$
16. ☐ $(A \implies B) \implies A$

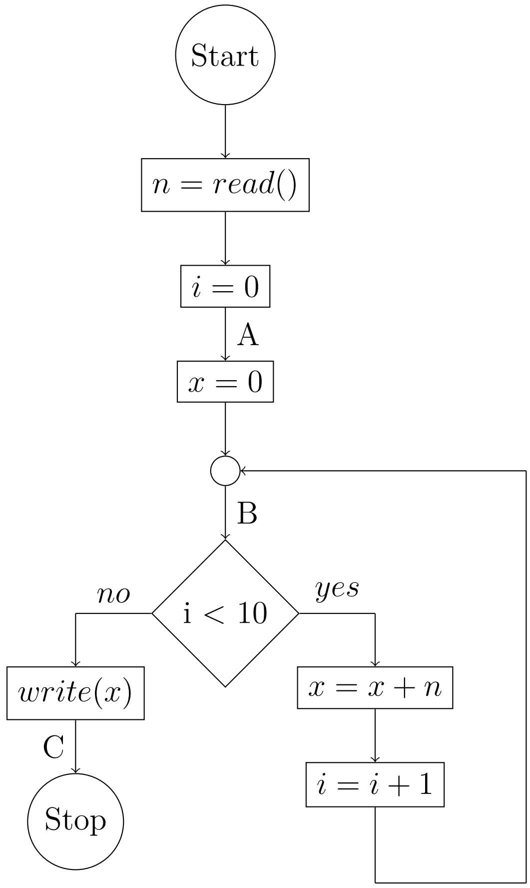
[1] https://www.in.tum.de/i02/lehre/wintersemester-1819/vorlesungen/functional-programming-and-verification/

[2] https://www.moodle.tum.de/course/view.php?id=44932

[3] https://piazza.com/tum.de/fall2018/in0003/home

**Assignment 1.2 (L) Assertions**

Consider the following control flow graph:




1. Which of the following assertions hold at point $A$?

   (a) $\boxtimes$ $i \geq 0$
   (b) $\square$ $x = 0$
   (c) $\square$ $i \leq 10 \wedge x \neq 0$
   (d) $\boxtimes$ $true$
   (e) $\boxtimes$ $i = 0$
   (f) $\square$ $x = i$

2. Which of the following assertions hold at point $B$?

   (a) $\square$ $x = 0 \wedge i = 0$
   (b) $\square$ $x = i$
   (c) $\square$ $i < x$
   (d) $\boxtimes$ $0 \leq i \leq 10$
   (e) $\square$ $i \geq 0 \wedge x \geq 0$
   (f) $\boxtimes$ $n = 1 \implies x = i$

3. Which of the following assertions hold at point $C$?

   (a) $\boxtimes$ $i \geq 0$
   (b) $\boxtimes$ $i = 10$
   (c) $\boxtimes$ $i > 0$
   (d) $\square$ $x \neq n$
   (e) $\boxtimes$ $x = 10n$
   (f) $\boxtimes$ $x = i * n \wedge i = 10$

**Assignment 1.3 (L) The Strong and the Weak**

Again consider the assertions that hold at point $C$ of assignment 1.2. Discuss the following questions:

1. When annotating the control flow graph, can you say that one of the given assertions is "better" than the others?
2. Can you arrange the given assertions in a meaningful order?
3. How can you define a *stronger than* relation formally?
4. How do *true* and $false$ fit in and what is their meaning as an assertion?
5. What are the strongest assertions that still hold at $A, B$ and $C$?

**Suggested Solution 1.3**

1. When we want to make a statement about the state (the values of variables) at a program point, the "best" assertion is the one that gives the most precise description of the values that are actually possible in the program. Therefore, an assertion $i = 10$ is "better" than $i \geq 0$, because the information it provides is more precise. We thus say that $i = 10$ is *stronger* than $i \geq 0$. Intuitively, $x = i * n \wedge i = 10$ is the strongest among the given assertions.

2. Following the above argument, we can at least say the following:

   - $x = i * n \wedge i = 10$ stronger than $i = 10$ stronger than $i > 0$ stronger than $i \geq 0$
   - $x = i * n \wedge i = 10$ stronger than $x = 10n$

   However, one cannot say that $i = 10$ is *stronger* than $x = 10n$ or vice versa, since the information they provide is not "better" or "worse", but just different or incomparable.

3. In some sence, an assertion $A$ is *stronger* than an assertion $B$, if $A$ extends or includes the information of $B$. In other words, whenever $A$ is satisfied, $B$ is satisfied as well. Hence,

   $$A \textit{ stronger than } B := A \implies B$$

   is a partial order (reflexive, transitive and anti-symmetric). We can now rewrite the above statements:

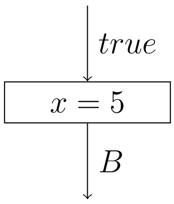
   - $(x = i * n \wedge i = 10 \implies i = 10) \wedge (i = 10 \implies i > 0) \wedge (i > 0 \implies i \geq 0)$
   - $x = i * n \wedge i = 10 \implies x = 10n$
   - Neither $i = 10 \implies x = 10n$, nor $i = 10 \impliedby x = 10n$

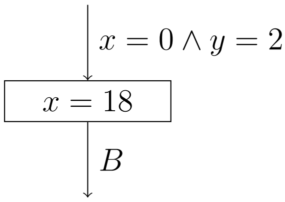
4. Since $false \implies A$ and $A \implies true$ hold for all $A$, $false$ and $true$ are the strongest and weakest assertions, respectively. While *true* is always satisfied, there is no assignment of variable values that can ever satisfy $false$. The only way for a program not to violate an assertion $false$ is to never reach the respective program point.

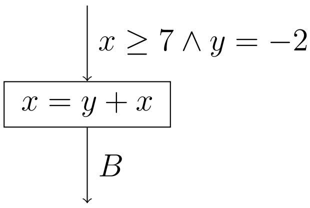
5. For $A$ and $C$ the strongest assertions are already given with $i = 0$ and $x = i * n \wedge i = 10$, respectively. However, for point $B$ we can still find a stronger assertion: $x = i * n \wedge 0 \leq i \leq 10$

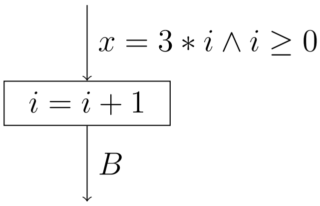
### Assignment 1.4 (L) Strongest Postconditions

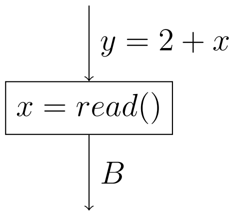
Using our understanding of strong and weak assertions, it is now possible to look at statements individually and, given an assertion $A$ that holds before a statement $s$, find the strongest assertion $B$ that holds after the statement. We call $B$ the *strongest postcondition* (of $A$ at statement $s$).
For each of the following statements, find the *strongest postcondition* $B$:

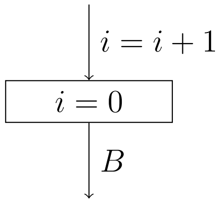
1.


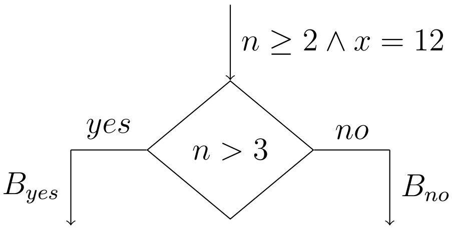

2.


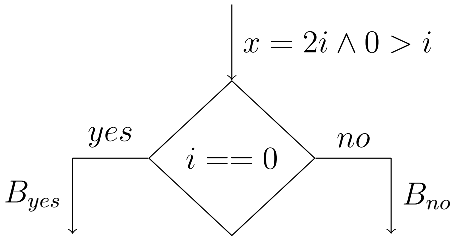

3.


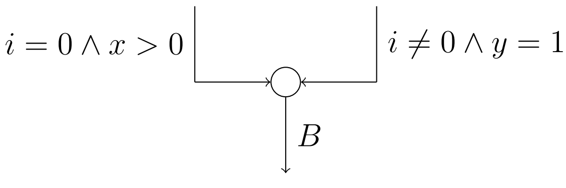

4.



5.



6.



7.



8.



9.



### Suggested Solution 1.4

1. $B :\equiv x = 5$
2. $B :\equiv x = 18 \wedge y = 2$
3. $B :\equiv x \geq 5 \wedge y = -2$
4. $B :\equiv x = 3(i - 1) \wedge i \geq 1$
5. $B :\equiv true$

6. $B :\equiv false$

7. $B_{yes} :\equiv n > 3 \land x = 12$ and $B_{no} :\equiv (n = 2 \lor n = 3) \land x = 12$

8. $B_{yes} :\equiv false$ and $B_{no} :\equiv x = 2i \land 0 > i$

9. $B :\equiv (i = 0 \land x > 0) \lor (i \neq 0 \land y = 1)$
   or $B :\equiv (i = 0 \implies x > 0) \land (i \neq 0 \implies y = 1)$

**Assignment 1.5 (H) Proving Ground: Implications** [4 Points]

Prove or refute the following claims. Whenever you rewrite any terms, annotate the rule that was used.

1. It exists an $A$ such that
   - $x < 0 \land y = 2 \implies A$ and
   - $A \implies x < -2 \land y > 0$

   are both satisfied.

2. No $A$ and $B$ exist such that
   - $A \implies B \land x > 2$,
   - $B \implies x > 5$ and
   - $x > 2 \implies A$

   hold.

3. $(A \land B \implies C) \impliedby B \lor C \equiv A \land \neg C \implies \neg B$

4. $(x < 0 \implies A) \land (\neg A \implies x < -2) \implies (y = 3 \implies A) \equiv true$

**Suggested Solution 1.5**

1. Such an $A$ does not exist. We prove by contradiction. Assume such an $A$ exists, then, due to transitivity of implication, $x < 0 \land y = 0 \implies x < -2 \land y > 0$ must hold. But this is not the case, so our assumption is wrong and such an $A$ does not exist. □

2. This statement is true. We rewrite the first implication as $(A \implies B) \land (A \implies x > 2)$. Now, we again prove by contradiction: Assume such an $A$ and $B$ exist, then again due to transitivity of implication $(x > 2 \implies A \implies B \implies x > 5)$, $x > 2$ has to imply $x > 5$, which is wrong and so is our initial assumption. □

3. Simply rewrite one side into the other using the well known rules:

$$
\begin{aligned}
&(A \land B \implies C) \impliedby B \lor C && \textit{(implication)}\\
\equiv&(A \land B \implies C) \lor \neg(B \lor C) && \textit{(De Morgan)}\\
\equiv&(A \land B \implies C) \lor \neg B \land \neg C && \textit{(implication)}\\
\equiv&\neg(A \land B) \lor C \lor (\neg B \land \neg C) && \textit{(De Morgan)}\\
\equiv&\neg A \lor \neg B \lor C \lor (\neg B \land \neg C) && \textit{(absorption)}\\
\equiv&\neg A \lor \neg B \lor C && \textit{(reorder)}\\
\equiv&\neg A \lor C \lor \neg B && \textit{(De Morgan)}\\
\equiv&\neg(A \land \neg C) \lor \neg B && \textit{(implication)}\\
\equiv&(A \land \neg C) \implies \neg B
\end{aligned}
$$

□

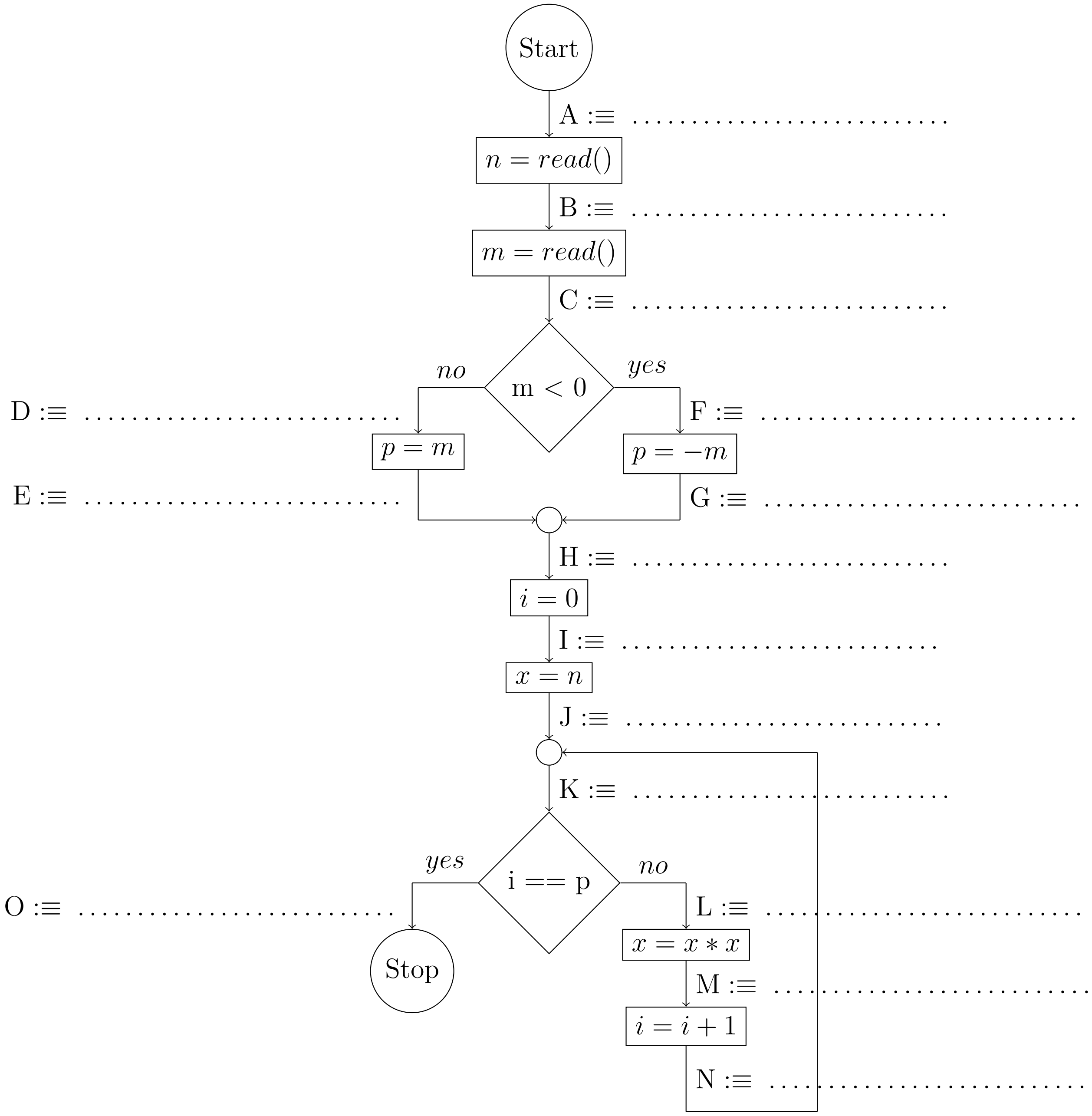
4. Simply rewrite one side into the other using the well known rules:

$$
\begin{aligned}
&(x < 0 \implies A) \land (\neg A \implies x < -2) \implies (y = 3 \implies A) && \textit{(transposition)}\\
\equiv&(x < 0 \implies A) \land (x \geq -2 \implies A) \implies (y = 3 \implies A) && \textit{(implications)}\\
\equiv&(x < 0 \lor x \geq -2 \implies A) \implies (y = 3 \implies A) && \textit{(tautology)}\\
\equiv&A \implies (y = 3 \implies A) && \textit{(implications)}\\
\equiv&\neg A \lor y \neq 3 \lor A && \textit{(tautology)}\\
\equiv&\texttt{true}
\end{aligned}
$$

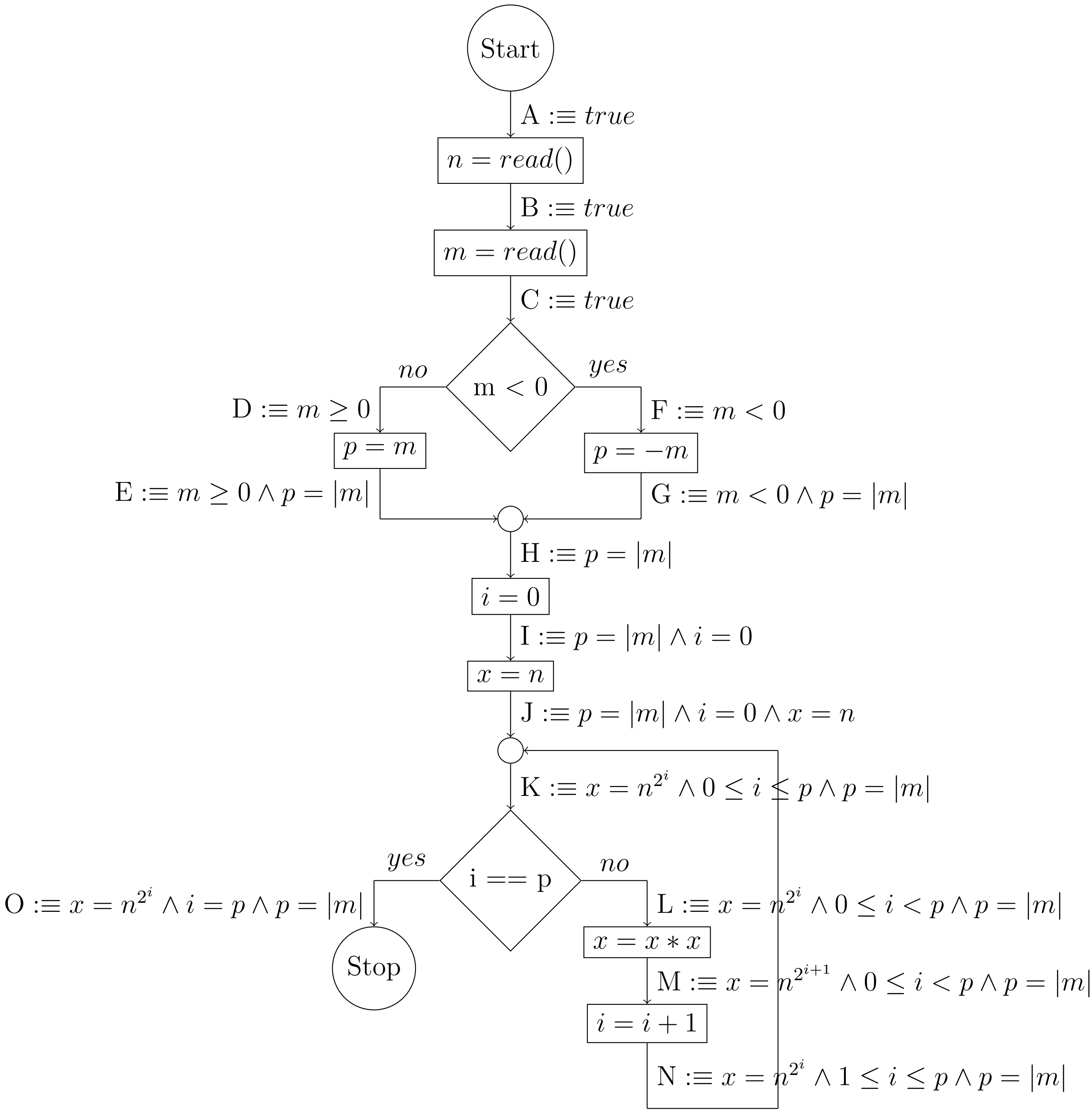
□

**Assignment 1.6 (H) Powerful Assertions** [7.5 Points]

In the following control flow graph, find the strongest assertions $A - O$ that hold at the correspondingly labeled edge:

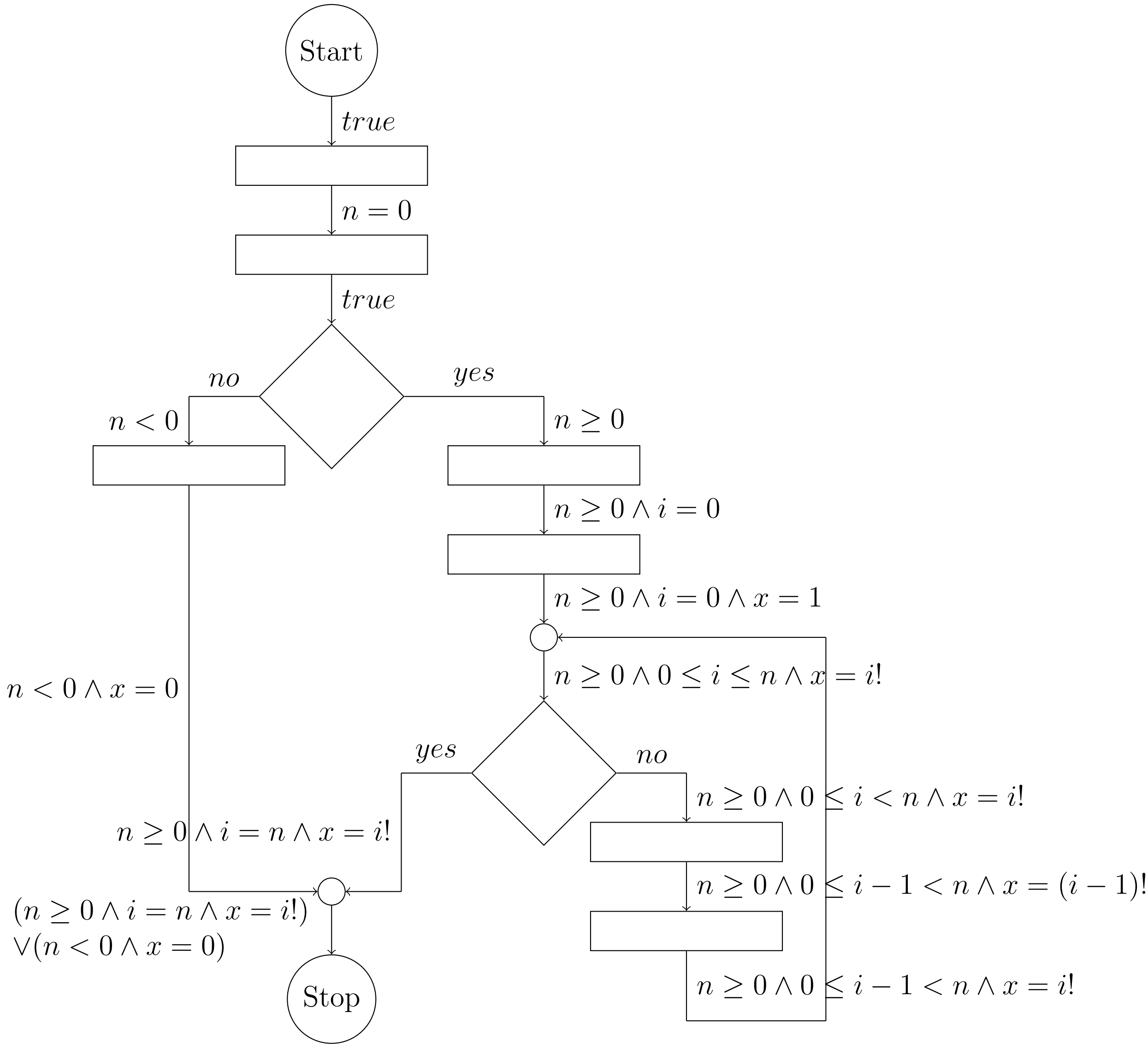



**Suggested Solution 1.6**

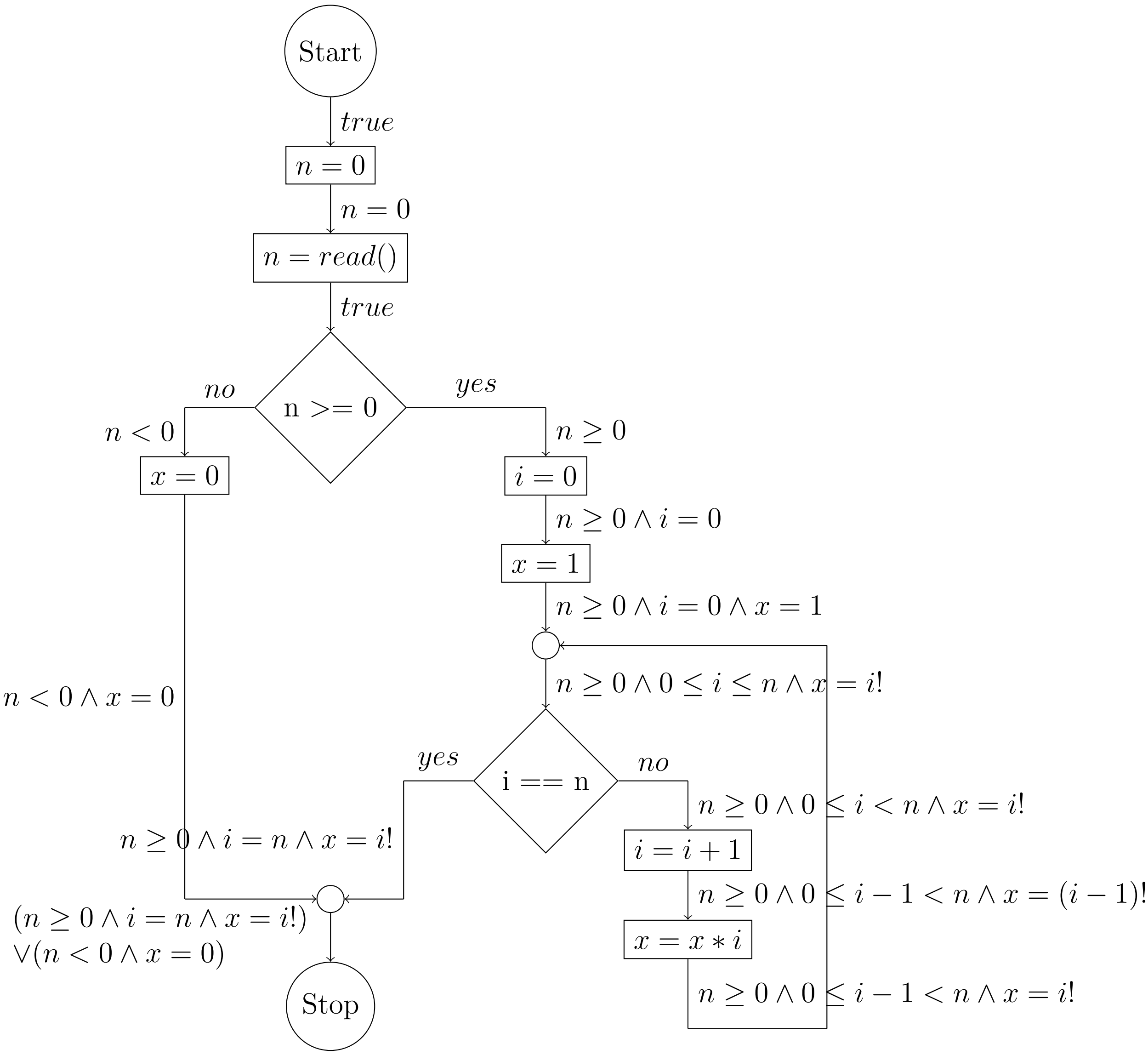

Start
A :≡ $true$
$n = read()$
B :≡ $true$
$m = read()$
C :≡ $true$
m < 0
no
yes
D :≡ $m \geq 0$
F :≡ $m < 0$
$p = m$
$p = -m$
E :≡ $m \geq 0 \wedge p = |m|$
G :≡ $m < 0 \wedge p = |m|$
H :≡ $p = |m|$
$i = 0$
I :≡ $p = |m| \wedge i = 0$
$x = n$
J :≡ $p = |m| \wedge i = 0 \wedge x = n$
K :≡ $x = n^{2^i} \wedge 0 \leq i \leq p \wedge p = |m|$
i == p
yes
no
O :≡ $x = n^{2^i} \wedge i = p \wedge p = |m|$
L :≡ $x = n^{2^i} \wedge 0 \leq i < p \wedge p = |m|$
Stop
$x = x * x$
M :≡ $x = n^{2^{i+1}} \wedge 0 \leq i < p \wedge p = |m|$
$i = i + 1$
N :≡ $x = n^{2^i} \wedge 1 \leq i \leq p \wedge p = |m|$

**Assignment 1.7 (H) Lost Statements** [4.5 Points]

The following control flow graph's edges are annotated with the strongest assertions that hold at the respective program point. Find the statements and conditions of the program.




**Suggested Solution 1.7**


Start
true
$n = 0$
$n = 0$
$n = read()$
true
n >= 0
no
yes
$n < 0$
$n \geq 0$
$x = 0$
$i = 0$
$n \geq 0 \land i = 0$
$x = 1$
$n \geq 0 \land i = 0 \land x = 1$
$n < 0 \land x = 0$
$n \geq 0 \land 0 \leq i \leq n \land x = i!$
yes
i == n
no
$n \geq 0 \land 0 \leq i < n \land x = i!$
$n \geq 0 \land i = n \land x = i!$
$i = i + 1$
$n \geq 0 \land 0 \leq i - 1 < n \land x = (i-1)!$
$(n \geq 0 \land i = n \land x = i!)$
$\lor (n < 0 \land x = 0)$
$x = x * i$
$n \geq 0 \land 0 \leq i - 1 < n \land x = i!$
Stop

**Assignment 1.8 (H) Return of the Penguins** [4 Points]

One day, while wandering around in the furthest corners of the university to find your exercise room, you walk by some laboratory, where a small penguin girl is tinkering on a strange device. After taking note of you watching her, she excitedly starts talking to you.

Hi there! I am Julia. I'm an informatics student! We penguins looooooove informatics! But now that I'm in third semester, I miss all my penguin friends from last year *sad*. There were so many other penguins, but now they are all gone .... *sad*. And that is why I built thiiiiiis time machine *excitedly pointing to the device*. Everything is ready for the first test, but before I can go back to the wonderful penguin times, I need to compute the correct parameters for the flux capacitor. I was planning to write a program for that, but sadly, I only learned how to write a compiler so far *sad*.

What did you say? You can help me with the programming and you already know about assertions?! Wow, this is aweeeesome! Then I can give you the formal specification of my program and you can write it for me: The program reads two inputs $a$ and $b$, that must not be modified within the program, once they are read. The program then computes the flux capacitor parameter $p$ such that

$$(a > b \implies p = -1) \wedge (a \leq b \implies (a \geq 0 \implies p = \tfrac{b!}{a!} * \sum_{k=a}^{b} k) \wedge (a < 0 \implies p = \sum_{k=0}^{b} k))$$

holds at the end of the program. Can you help me?

Help Julia and implement this program in MiniJava. Inputs $a$ and $b$ must not be modified within the program once they are read from the user.

**Suggested Solution 1.8**

Julia is watching you fascinated as you think for a moment and write down the program:

```
a = read();
b = read();
p = -1
if(a <= b)
{
  if(a >= 0)
  {
    s = a;
    f = 1;
    i = a;
    while(i < b)
    {
```

```
      i = i + 1;
      s = s + i;
      f = f * i;
    }
    p = s * f;
  }
  else
  {
    i = 0;
    p = 0;
    while(i < b)
    {
      i = i + 1;
      p = p + i;
    }
  }
}
write(p);
```

You hand it to Julia. She immediately computes the necessary parameters and, while mumbling lots of technical stuff, enters them to a small keyboard next to the flux capacitor. Without saying much, Julia jumps onto the time machine, pulls a couple of levers and, under bursts of sparks erupting out of the machine, disappears, leaving nothing behind but a cloud of charged air. Stunned by what just happend, you stare into the cloud as it slowly vanishes. However, a feeling arises that you will see Julia again, soon...