

Dr. Liam O'Connor University of Edinburgh LFCS UNSW, Term 3 2020

### **Imperative Programming**

## Assignment Project Exam Help

### https://powcoder.com

**Definition** 

*Imperative programming* is where programs are described as a series of *statements* or commands to manipulate mutable *state* or cause externally observable *effects*.

States may take the form of a mapping from variable names to their values, or even a model of a CPU state with a memory model (for example, in an assembly language).

### The Old Days



Early microcomputer languages used a line numbering system with GO TO statements used to arrange control flow.

### Factorial Example in BASIC (1964)

Assignment Project Exam Help

```
10 N = 4

20 I = 0

34 ttps://powcoder.com 100

50 I = I + 1

60 M = M * I

70 GOTO 40

110 dg RWe Chat powcoder
```

### **Dijkstra** (1968)

# nt Project Exam Help k//powcoder.com

#### Godo Stalement Considered Harmen WCOde Isla Words and Phrases: go to statement, jump instruction. we c branch instruction, conditional clause, alternative clause, repettext itive clause, program intelligibility, program sequencing dyna CR Categories: 4.22, 5.23, 5.24

The structured programming movement brought in control structures to mainstream use, such as conditionals and loops. 

### Factorial Example in Pascal (1970)

```
program factorial;
Assignment Project Exam Help
               i : integer;
     https://pewcoder.com
            while (in an) dowcoder
             m := m * i;
            end:
            println(m);
```

### **Syntax**

Hoare Logic

We're going to specify a language Tinylmp, based on structured programming. The syntax estimated Presion ect Exam Help

We already know how to make unambiguous abstract syntax, so we will use concrete syntax in the rules for readability.

Grammar

### **Examples**

```
Exam Help
                       var m · var n · var i ·
          var i ·
                       m := 1; n := 1;
       https://powcoder.com
          m := 1:
                        var t \cdot t := m;
          while i < N do
       Add=WeChat powcoder
                       od
```

Types?

### Assignment Project Exam Help

https://powcoder.com



**Types?** We only have one type (int), so type checking is a wash.

### ScopA?ssignment Project Exam Help

https://powcoder.com

**Types?** We only have one type (int), so type checking is a wash.

Scop ? We in a post of the property of the pro

https://powcoder.com

**Types?** We only have one type (int), so type checking is a wash.

### Scope? Sweingerproper that variables are initialized before they are used! Anything Else? We have to check that variables are initialized before they are used!

https://powcoder.com

Types? We only have one type (int), so type checking is a wash.

### Scope? We have to check that variables are initialized before they are used!

https://powcoder.com





Types? We only have one type (int), so type checking is a wash.

### Scope? We have to check that variables are initialized before they are used!

https://powcoder.com

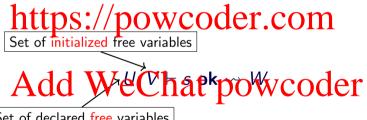
Add WeChatkpowcoder

Set of declared free variables

Types? We only have one type (int), so type checking is a wash.

### scopa?sysignment value jectre Exam Help

Anything Else? We have to check that variables are *initialized* before they are used!



Set of declared free variables

Types? We only have one type (int), so type checking is a wash.

### Scope? We have to check that variables are initialized before they are used!

https://powcoder.com
Set of initialized free variables

Add WeChatkpowcoder

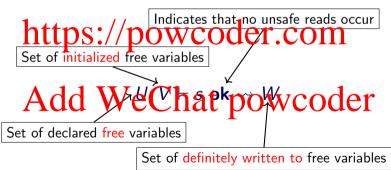
Set of declared free variables

Note:  $V \subseteq U$ 

Types? We only have one type (int), so type checking is a wash.

### scopa?sveignoment value jeetre Exam Help

Anything Else? We have to check that variables are initialized before they are used!



Note:  $V \subseteq U$ 



# $Assign{\color{red} \underline{ment \ Project \ Exam \ Help}}$

https://powcoder.com

### $Assignment Project Exam Help \\ \overline{U; V \vdash \text{skip ok} \rightsquigarrow \emptyset} U; V \vdash x := e \text{ ok} \rightsquigarrow$

https://powcoder.com

## $Assignment Project Exam Help \\ \overline{U; V \vdash \text{skip ok} \leadsto \emptyset}$

https://powcoder.com

### Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightarrow \emptyset$

https://powcoder.com

## 

https://powcoder.com

### Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightarrow \emptyset$ $U; V \vdash x := e \text{ ok} \rightarrow \{x\}$

https://powcoder.com

### Assignment Project Exem Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset \qquad U; V \vdash x := e \text{ ok} \rightsquigarrow \{x\}$

 $https \frac{y \cup \{y\}; V \vdash s \text{ ok } \rightarrow W}{y \not : v \not = \mathbf{ok} \rightarrow \mathbf{ok}}$ 

### Assignment Project Exem Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset \qquad U; V \vdash x := e \text{ ok} \rightsquigarrow \{x\}$

 $https \frac{y \cup \{y\}; V \vdash s \text{ ok } \rightarrow W}{y \not : V \not = \mathbf{ok} \rightarrow \mathbf{coder} \cup \mathbf{com}}$ 

### Assignment Project Exem Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset \qquad U; V \vdash x := e \text{ ok} \rightsquigarrow \{x\}$

 $https \frac{y \cup \{y\}; V \vdash s \text{ ok } \rightarrow W}{y \not : V \not = \mathbf{ok} \rightarrow \mathbf{coder} \subseteq \mathbf{com}}$ 

 $Add \overset{\text{if $e$ then $s_1$ else $s_2$ fi ok $\sim$}}{WeChat} powcoder$ 

### Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset$

```
\underset{\mathrm{FV}(e) \subset V}{\text{https}} \overset{\text{$V \cup \{y\}; $V \vdash s$ ok } \longrightarrow W}{\text{$W$ colorwise}}
```

 $Add \overset{\text{if $e$ then $s_1$ else $s_2$ fi ok $\sim$}}{WeChat} powcoder$ 

### Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset$

```
\frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https } \mathcal{U} : \mathcal{V} \text{poweder com}}
\text{FV}(e) \subseteq V \qquad U; V \vdash s_1 \text{ ok } \rightsquigarrow W_1
```

 $Add \overset{\text{U-V- if } e \text{ then } s_1 \text{ else } s_2 \text{ fi } \text{ ok } \leadsto \\ Add & We Chat powcoder \\$ 

# 

```
\frac{\mathsf{https}\, \mathcal{V} \cup \{y\}; \, V \vdash s \, \mathsf{ok} \rightsquigarrow W}{\mathsf{porw}\, \mathsf{coeler}, \mathsf{com}}
\mathrm{FV}(e) \subseteq V \qquad U; \, V \vdash s_1 \, \mathsf{ok} \rightsquigarrow W_1 \qquad U; \, V \vdash s_2 \, \mathsf{ok} \rightsquigarrow W_2}
U: V \vdash \mathsf{if}\, e \, \mathsf{then}\, s_1 \, \mathsf{else}\, s_2 \, \mathsf{fi}\, \mathsf{ok} \rightsquigarrow
```

 $Add^{\textit{U-V}} \overset{\text{if $e$ then $s_1$ else $s_2$ fi ok $\sim$}}{WeChat \ powcoder}$ 

```
\frac{\text{https}\, \overline{\mathcal{V}_{\cdot} \, \mathcal{V}_{\text{porw.coder}} \, \mathcal{W}_{\cdot} \, \mathcal{W}_{\text{porw.coder}} \, \mathcal{W}_{\text{total pow.coder}} \, \mathcal{W}_{\text{porw.coder}} \, \mathcal{
```

```
\frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{https}} \frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W_1}{\text{
```

```
\frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https} \ \mathcal{Y} : \mathcal{N} \text{porw coder}}
FV(e) \subseteq V \qquad U; V \vdash s_1 \text{ ok } \rightsquigarrow W_1 \qquad U; V \vdash s_2 \text{ ok } \rightsquigarrow W_2
\frac{U; V \vdash \text{ if } e \text{ then } s_1 \text{ else } s_2 \text{ fi ok } \rightsquigarrow W_1 \cap W_2}{\text{Add} FW} \text{powcoder}
U: V \vdash \text{ while } e \text{ do } s \text{ od ok } \rightsquigarrow
```

```
\frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https} \ \mathcal{Y} : \mathcal{N} \text{porw coder} \text{form}}
\text{FV}(e) \subseteq V \qquad U; V \vdash s_1 \text{ ok } \rightsquigarrow W_1 \qquad U; V \vdash s_2 \text{ ok } \rightsquigarrow W_2}
\frac{U; V \vdash \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi ok } \rightsquigarrow W_1 \cap W_2}{\text{How coder}}
\frac{U; V \vdash \text{while } e \text{ do } s \text{ od ok } \rightsquigarrow}{U; V \vdash \text{while } e \text{ do } s \text{ od ok } \rightsquigarrow}
```

```
\frac{V \cup \{y\}; V \vdash s \text{ ok } \rightsquigarrow W}{\text{https} \ \mathcal{Y} : V \not\models \text{ok} \qquad \mathcal{W}}
\text{FV(e)} \subseteq V \qquad U; V \vdash s_1 \text{ ok } \rightsquigarrow W_1 \qquad U; V \vdash s_2 \text{ ok } \rightsquigarrow W_2
\frac{U; V \vdash \text{ if } e \text{ then } s_1 \text{ else } s_2 \text{ fi ok } \rightsquigarrow W_1 \cap W_2}{\text{Add} \ \text{FV(e)} = \text{Chat} \vdash \text{powcoder}}
U: V \vdash \text{ while } e \text{ do } s \text{ od } \text{ok } \rightsquigarrow \emptyset
```

## Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset$

```
\begin{array}{c|c}
https & V \cup \{y\}; V \vdash s \text{ ok } \longrightarrow W \\
\hline
\text{FV(e)} \subseteq V & U; V \vdash s_1 \text{ ok } \longrightarrow W_1 & U; V \vdash s_2 \text{ ok } \longrightarrow W_2
\end{array}
```

 $Add_{FW} \stackrel{\text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi } \text{ok} \rightsquigarrow W_1 \cap W_2}{\text{FW} \text{echat } \text{poweoder}}$ 

U;  $V \vdash$ while e do s od **ok**  $\leadsto \emptyset$ 

U;  $V \vdash s_1$ ;  $s_2$  ok  $\rightsquigarrow$ 

```
https \cancel{y} : \bigvee pewceder we on
FV(e) \subseteq V U: V \vdash s_1 \text{ ok } \rightsquigarrow W_1 U: V \vdash s_2 \text{ ok } \rightsquigarrow W_2
   Add_{FW}^{U_1V_1 - \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi } \text{ok} \rightsquigarrow W_1 \cap W_2} \\ Add_{FW} e Chat + powcoder
                           U: V \vdash \text{while } e \text{ do } s \text{ od } \mathbf{ok} \leadsto \emptyset
        U: V \vdash s_1 \text{ ok} \rightsquigarrow W_1
                               U: V \vdash s_1: s_2 \text{ ok} \leadsto
```

#### **Static Semantics Rules**

# Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset$

$$\frac{\mathbf{https}\, \mathcal{U} \cup \{y\}; \, V \vdash s \, \mathbf{ok} \rightsquigarrow W}{\mathbf{https}\, \mathcal{U} \wedge \mathbf{porweader} \in \mathbf{COM}}$$

$$\underline{FV(e) \subseteq V \quad U; \, V \vdash s_1 \, \mathbf{ok} \rightsquigarrow W_1 \quad U; \, V \vdash s_2 \, \mathbf{ok} \rightsquigarrow W_2}$$

$$\frac{U; \, V \vdash \text{ if } e \, \text{then } s_1 \, \text{else } s_2 \, \text{fi} \, \mathbf{ok} \rightsquigarrow W_1 \cap W_2}{\mathbf{Add}\, \underbrace{FV(V) \in \mathbf{Chat} \vdash \mathbf{poweOder}}_{U; \, V \vdash \text{ while } e \, \text{do } s \, \text{od} \, \mathbf{ok} \rightsquigarrow \emptyset}$$

$$\underline{U; \, V \vdash s_1 \, \mathbf{ok} \rightsquigarrow W_1 \quad U; \, (V \cup W_1) \vdash s_2 \, \mathbf{ok} \rightsquigarrow W_2}$$

$$\underline{U; \, V \vdash s_1 : s_2 \, \mathbf{ok} \rightsquigarrow}$$

#### **Static Semantics Rules**

# Assignment Project Exeam Help $U; V \vdash \text{skip ok} \rightsquigarrow \emptyset$

$$\frac{\mathbf{https}\, \mathcal{U} \cup \{y\}; \, V \vdash s \, \mathbf{ok} \rightsquigarrow W}{\mathbf{https}\, \mathcal{U} \wedge \mathbf{porweader} \in \mathbf{COM}}$$

$$\underline{FV(e) \subseteq V \quad U; \, V \vdash s_1 \, \mathbf{ok} \rightsquigarrow W_1 \quad U; \, V \vdash s_2 \, \mathbf{ok} \rightsquigarrow W_2}$$

$$\frac{U; \, V \vdash \text{ if } e \, \text{then } s_1 \, \text{else } s_2 \, \text{fi} \, \mathbf{ok} \rightsquigarrow W_1 \cap W_2}{\mathbf{Add}\, \underbrace{FWV \in \mathbf{hat} \vdash \mathbf{poweoder}}_{U; \, V \vdash \text{ while } e \, \text{do } s \, \text{od} \, \mathbf{ok} \rightsquigarrow \emptyset}$$

$$\underline{U; \, V \vdash s_1 \, \mathbf{ok} \rightsquigarrow W_1 \quad U; \, (V \cup W_1) \vdash s_2 \, \mathbf{ok} \rightsquigarrow W_2}$$

$$\underline{U; \, V \vdash s_1 : s_2 \, \mathbf{ok} \rightsquigarrow W_1 \cup W_2}$$

We will use big-step operational semantics. What are the sets of evaluable expressions and Aussian Project Exam Help

https://powcoder.com

We will use big-step operational semantics. What are the sets of evaluable expressions and Ausseignment Project Exam Help Evaluable Expressions:

https://powcoder.com

We will use big-step operational semantics. What are the sets of evaluable expressions and Aussian Project Exam Help Evaluable Expressions: A pair containing a statement to execute and a state  $\sigma$ .

Values:

https://powcoder.com

We will use big-step operational semantics. What are the sets of evaluable expressions and Aussian Project Exam Help Evaluable Expressions: A pair containing a statement to execute and a state  $\sigma$ .

Values: The final state that results from executing the statement.

https://powcoder.com

We will use big-step operational semantics. What are the sets of evaluable expressions and Ausseignment Project Exam Help

Evaluable Expressions: A pair containing a statement to execute and a state  $\sigma$ .

Values: The final state that results from executing the statement.

### https://powcoder.com

A *state* is a mutable mapping from variables to their values. We use the following notation:

- To read a variable from the state q, wt write of two coder
  To update an existing variable x to have value v inside the state σ, we write
- To update an existing variable x to have value v inside the state  $\sigma$ , we write  $(\sigma : x \mapsto v)$ .
- To extend a state  $\sigma$  with a new, previously undeclared variable x, we write  $\sigma \cdot x$ . In such a state, x has undefined value.

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture  $\frac{1}{2}$  Project  $\frac{1}{2}$  Exam  $\frac{1}{2}$  Help

 $\overline{(\sigma,\mathtt{skip}) \Downarrow \sigma}$ 

https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture

# ike iAthe previous lecture ent Project Exam Help

 $\overline{(\sigma,\mathtt{skip}) \Downarrow \sigma} \quad \overline{(\sigma_1,s_1;s_2) \Downarrow}$ 

https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in the previous lecture ent Project Exam Help  $\frac{(\sigma, skip) \Downarrow \sigma}{(\sigma, skip) \Downarrow \sigma}$ 

https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in Athen previous lecture ent Project Exam Help 
$$(\sigma, \text{skip}) \downarrow \sigma$$
  $(\sigma_1, s_1; s_2) \downarrow$ 

https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much



https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in Athen previous lecture ent Project Exam Help 
$$(\sigma, \text{skip}) \downarrow \sigma$$
  $(\sigma_1, s_1; s_2) \downarrow \sigma_3$ 

https://powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in Athen previous lecture ent Project Exam Help 
$$(\sigma, \text{skip}) \downarrow \sigma$$
  $(\sigma_1, s_1; s_2) \downarrow \sigma_3$ 

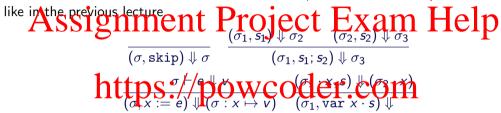
https: //powcoder.com

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in the previous lecture ent 
$$Project$$
  $Examt{Exam}$   $Helptoolean formula  $F(\sigma_1,s_1) \downarrow \sigma_2$   $F(\sigma_2,s_2) \downarrow \sigma_3$   $F(\sigma_3,s_1;s_2) \downarrow \sigma_3$   $F(\sigma_3,s_1;s_2) \downarrow \sigma_3$$ 

$$\operatorname{https://powcoder.com}_{(a_{\times}:=e)}/\operatorname{powcoder.com}_{(\sigma_{1},\operatorname{var}\times\cdot s)}$$

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much



We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in the previous lecture ent 
$$Project$$
  $Exam_{(\sigma, skip) \ \psi \ \sigma}$   $Project$   $Exam_{(\sigma_1, s_1; s_2) \ \psi \ \sigma_3}$   $Project$   $Project$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in the previous lecture ent Project Exam Help 
$$(\sigma, \text{skip}) \downarrow \sigma$$
  $(\sigma, s_1; s_2) \downarrow \sigma_3$ 

$$\operatorname{https:}_{(\sigma_1, \operatorname{var} \times \cdot s)}^{\sigma//powc} \circ \operatorname{der}_{(\sigma_1, \operatorname{var} \times \cdot s)}^{\circ/powc} \circ \operatorname{der}_{(\sigma_1, \operatorname{var} \times \cdot s)}^{\circ/powc} \circ \operatorname{der}_{(\sigma_2, \operatorname{var} \times s)}^{\circ/powc} \circ \operatorname{der}_{$$

### AddWeChatpowcoder

 $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \Downarrow$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

$$\underset{(\sigma,\,\mathtt{skip})\,\psi\,\sigma}{\text{like in Athe previous lecture ent}} \, \underset{(\sigma,\,\mathtt{skip})\,\psi\,\sigma}{\text{Project }} \, \underset{(\sigma_1,\,s_1;\,s_2)\,\psi\,\sigma_3}{\text{Exam}} \, \, \text{Help}$$

$$\operatorname{https://powcoder}_{(\sigma_1, \operatorname{var} x \cdot s) \downarrow \sigma_2}$$

 $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0$ 

Add Weehat powcoder

 $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in Athen previous lecture ent  $P_{(\sigma_1, s_1)}$  ject  $E_{(\sigma_2, s_2)}$  Help  $(\sigma, \text{skip}) \Downarrow \sigma$   $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$  $\operatorname{https://powcoder.com}_{(\sigma_1, \operatorname{var} x \cdot s) \cup \sigma_2}$  $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0 \qquad (\sigma_1, s_1) \Downarrow \sigma_2$ 

Add Weehat powcoder

 $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow \downarrow$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the provious lecture.

like in the previous lecture at Project Exam Help 
$$\frac{(\sigma, \operatorname{skip}) \Downarrow \sigma}{(\sigma, \operatorname{skip}) \Downarrow \sigma} \xrightarrow{(\sigma_1, s_1; s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1, s$$

 $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture v = v.

like in the previous lecture ent Project Exam Help 
$$\frac{(\sigma_1,s_1) \Downarrow \sigma_2}{(\sigma,\operatorname{skip}) \Downarrow \sigma} \xrightarrow{(\sigma_1,s_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1) \Downarrow \sigma_2} \operatorname{https}_{(\sigma_1,x_1) \Downarrow \sigma_2} \operatorname{Add}_{(Vifether late power oder}_{(\sigma_1,\operatorname{if}\ e\ \operatorname{then}\ s_1\ \operatorname{else}\ s_2\ \operatorname{fi}) \Downarrow}$$

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture.

like in the previous lecture ent Project Exam Help 
$$\frac{(\sigma_1,s_1) \Downarrow \sigma_2}{(\sigma,\operatorname{skip}) \Downarrow \sigma} \xrightarrow{(\sigma_1,s_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1;s_2) \Downarrow \sigma_3} \operatorname{https}_{(\sigma_1,x_1) \Downarrow \sigma_2} \operatorname{https}_{(\sigma_1,x_1) \Downarrow \sigma_2} \operatorname{Add}_{(\bullet Vifer helitates of (\sigma_1,s_2) \Downarrow \sigma_2} \operatorname{Add}_{(\sigma_1,\operatorname{if}\ e\ then\ s_1\ else\ s_2\ fi) \Downarrow \sigma_2}$$

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture v = v.

like in Athen previous lecture ent  $P_{\sigma_1, s_1}$  ject  $E_{s_2, s_2}$  Help  $(\sigma, \mathtt{skip}) \Downarrow \sigma$  $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$  $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0$  $(\sigma_1, s_1) \Downarrow \sigma_2$  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \Downarrow \sigma_2$ 

 $(\sigma_1, \text{while } e \text{ do } s \text{ od}) \downarrow$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture  $v \vdash v$  for arithmetic expressions, much

like in Athense in Project Exam Help  $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$  $(\sigma, \mathtt{skip}) \Downarrow \sigma$  $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0$  $(\sigma_1, s_1) \Downarrow \sigma_2$ Add Weet at the detail of the  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \Downarrow \sigma_2$ 

 $\sigma_1 \vdash e \Downarrow 0$ 

 $(\sigma_1, \text{while } e \text{ do } s \text{ od}) \downarrow$ 

 $(\sigma_1, \text{while } e \text{ do } s \text{ od})$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture  $v \vdash v$  to  $v \vdash v$  to  $v \vdash v$  to  $v \vdash v$ .

like in Athense in Project Exam Help  $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$  $(\sigma, \mathtt{skip}) \Downarrow \sigma$  $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0$  $(\sigma_1, s_1) \Downarrow \sigma_2$ Add Weet at the detail of the  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \Downarrow \sigma_2$ 

$$\sigma_1 \vdash e \Downarrow 0$$

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture  $v \in A$ . The previous lecture  $v \in A$  and  $v \in A$  anoth  $v \in A$  and  $v \in A$  and  $v \in A$  and  $v \in A$  and  $v \in A$  and

like in Athense in Project Exam Help  $(\sigma, \mathtt{skip}) \Downarrow \sigma$  $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$ https://powcoder.com/ $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0 \qquad (\sigma_1, s_1) \Downarrow \sigma_2$ Add Wie fination wooder  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \Downarrow \sigma_2$  $\sigma_1 \vdash e \Downarrow v \quad v \neq 0$  $\sigma_1 \vdash e \Downarrow 0$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much

like in Athen previous lecture ent  $P_{(\sigma_1, s_1)}$  ject  $E_{(\sigma_2, s_2)}$  Help  $(\sigma, \text{skip}) \Downarrow \sigma$   $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$ https://powcoder.com/ $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0 \qquad (\sigma_1, s_1) \Downarrow \sigma_2$ Add Went at the details and th  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow \sigma_2$  $\sigma_1 \vdash e \Downarrow v \quad v \neq 0$  $\sigma_1 \vdash e \downarrow \downarrow 0$  $(\sigma_1,s) \Downarrow \sigma_2$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture v = v.

like in Athen previous lecture ent  $P_{\sigma_1, s_1}$  ject  $E_{s_2, s_2}$  Help  $(\sigma, \text{skip}) \Downarrow \sigma$   $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$ https:/powcoder.com $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0 \qquad (\sigma_1, s_1) \Downarrow \sigma_2$ Add Weetat owcoder  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow \sigma_2$  $\sigma_1 \vdash e \Downarrow v \quad v \neq 0$  $(\sigma_1, s) \Downarrow \sigma_2 \quad (\sigma_2, \text{while } e \text{ do } s \text{ od}) \Downarrow \sigma_3$  $\sigma_1 \vdash e \Downarrow 0$  $(\sigma_1, \text{while } e \text{ do } s \text{ od})$  $(\sigma_1, \text{while } e \text{ do } s \text{ od}) \Downarrow \sigma_1$ 

We will assume we have defined a relation  $\sigma \vdash e \Downarrow v$  for arithmetic expressions, much like in the previous lecture v = v.

like in Athen previous lecture ent  $P_{\sigma_1, s_1}$  ject  $E_{s_2, s_2}$  Help  $(\sigma, \text{skip}) \Downarrow \sigma$   $(\sigma_1, s_1; s_2) \Downarrow \sigma_3$ https:/powcoder.com $\sigma_1 \vdash e \Downarrow v \qquad v \neq 0 \qquad (\sigma_1, s_1) \Downarrow \sigma_2$ Add Weetat owcoder  $(\sigma_1, \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi}) \downarrow \sigma_2$  $\sigma_1 \vdash e \Downarrow v \quad v \neq 0$  $(\sigma_1, s) \Downarrow \sigma_2 \quad (\sigma_2, \text{while } e \text{ do } s \text{ od}) \Downarrow \sigma_3$  $\sigma_1 \vdash e \Downarrow 0$  $(\sigma_1, \text{while } e \text{ do } s \text{ od}) \Downarrow \sigma_3^{\circ}$  $(\sigma_1, \text{while } e \text{ do } s \text{ od}) \downarrow \sigma_1$ 

#### **Hoare Logic**

To give you a taste of axiomatic semantics, and also how formal verification works, we are going to define what's called a floare Logic for Tirving to allow up to properties of our program. We write a Hoare triple judgement as:

### https://powcoder.com

Where  $\varphi$  and  $\psi$  are logical formulae about state variables, called assertions, and s is a statement. This triple states that of the statement is Swessfully evaluates from a starting state satisfying the precondition  $\varphi$ , then the result state will satisfy the postcondition  $\psi$ :

$$\varphi(\sigma) \wedge (\sigma, s) \Downarrow \sigma' \Rightarrow \psi(\sigma')$$

#### **Proving Hoare Triples**

```
To prove a Hoare triple like: Project Exam Help
                    i := 0:
        https://paw.coder.com
                     i := i + 1:
                     m := m \times i
        Add Wechat powcoder
```

It is undesirable to look at the operational semantics derivations of this whole program to compute what the possible final states are for a given input state. Instead we shall define a set of rules to prove Hoare triples directly (called *a proof calculus*).

# Assignment Project Exam Help

```
\frac{ \underset{(\sigma_1, s_1) \ \downarrow \ \sigma_2}{\overset{(\sigma_1, s_1) \ \downarrow \ \sigma_2}{\overset{(\sigma_1, s_1) \ \downarrow \ \sigma_3}{\overset{(\sigma_1, s_1) \ \hookrightarrow \ \sigma_3}{\overset
```

## Assignment Project Exam Help

```
\frac{ \underset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{2}}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{2}}{\underset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma_{1}, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{2}, s_{2}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ \sigma_{3}}{\underset{(\sigma, x := \ e) \ \downarrow \ (\sigma : x \mapsto v)}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1}) \ \downarrow \ (\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{1})}{\underset{(\sigma_{1}, s_{1})}{\overset{(\sigma_{1}, s_{
```

→□▶→□▶→□▶→□▶ □ り○○

## Assignment Project Exam Help

```
\frac{(\sigma_{1},s_{1}) \Downarrow \sigma_{2} \qquad (\sigma_{2},s_{2}) \Downarrow \sigma_{3}}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}}}} \underbrace{-\overset{\{\varphi\} s_{1};s}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}}}} \underbrace{-\overset{\{\varphi\} s_{1};s}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}{\underset{\sigma \vdash e \Downarrow v}{\underset{\sigma \vdash e \Downarrow v}{\overset{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}}{\underset{\sigma \vdash e \Downarrow v}{\underset{\sigma \vdash
```

# Assignment Project Exam Help $(\sigma_1, s_1) \Downarrow \sigma_2$ $(\sigma_2, s_2) \Downarrow \sigma_3$ $\{\varphi\} s_1 \{\alpha\}$ $\{\alpha\}$ $\{\alpha\}$ https://powcoder.com $\overset{(\sigma, x := e) \ \downarrow \ (\sigma : x \mapsto v)}{\text{Add WeChat powcoder}}$

# Assignment Project (Exam Help $\frac{(\sigma_{1},s_{1}) \Downarrow \sigma_{2} \quad (\sigma_{2},s_{2}) \Downarrow \sigma_{3}}{(\sigma_{1},s_{1}) \Downarrow \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{1} \{\alpha\} \quad \{\alpha\} s_{2} \{\psi\}}_{(\sigma_{1},s_{1}) \circlearrowleft \sigma_{3}} \underbrace{\{\varphi\} s_{2} \{\psi\}}_{(\sigma_{1},s_{2}) \end{dcases} \sigma_{3}}$

## Assignment Project Exam Help

```
(\sigma_1, s_1) \Downarrow \sigma_2 \qquad (\sigma_2, s_2) \Downarrow \sigma_3 \quad \{\varphi\} \ s_1 \ \{\alpha\} \qquad \{\alpha\} \ s_2 \ \{\psi\}
                                                                                                                                                               h^{(\sigma_1, s_1)} h^{s_2} h^{\sigma_3} h^{\sigma_3}
```

$$(\sigma, x := e) \Downarrow (\sigma : x \mapsto v)$$
  $\{\varphi[x := e]\} \ x := e \ \{\varphi\}$ 

# $\overbrace{Add}^{(\sigma,x:=e) \ \Downarrow \ (\sigma:x\mapsto v)} \overbrace{Add}^{\{\varphi[x:=e]\} \ x:=e \ \{\varphi\}}$

```
\{\varphi\} if e then s_1 else s_2 fi \{\psi\}
                                               \{\varphi\} while e do s od
```

## Assignment Project Exam Help

```
(\sigma_1, s_1) \Downarrow \sigma_2 (\sigma_2, s_2) \Downarrow \sigma_3 \{\varphi\} s_1 \{\alpha\} \{\alpha\} s_2 \{\psi\}
                                                                                                                                                              h^{(\sigma_1, s_1)} h^{s_2} h^{\sigma_3} h^{\sigma_3}
```

# $\overbrace{Add}^{(\sigma,x:=e) \ \Downarrow \ (\sigma:x\mapsto v)} \overbrace{Add}^{\{\varphi[x:=e]\} \ x:=e \ \{\varphi\}}$

```
\{\varphi \wedge e\} s_1 \{\psi\}
\{\varphi\} if e then s_1 else s_2 fi \{\psi\} \{\varphi\} while e do s od
```

## Assignment Project Exam Help

```
(\sigma_1, s_1) \Downarrow \sigma_2 (\sigma_2, s_2) \Downarrow \sigma_3 \{\varphi\} s_1 \{\alpha\} \{\alpha\} s_2 \{\psi\}
                                                                                                                                                              h^{(\sigma_1, s_1)} h^{s_2} h^{\sigma_3} h^{\sigma_3}
```

# $\overbrace{Add}^{(\sigma,x:=e) \ \Downarrow \ (\sigma:x\mapsto v)} \overbrace{Add}^{\{\varphi[x:=e]\} \ x:=e \ \{\varphi\}}$

$$\frac{\{\varphi \land e\} \ s_1 \ \{\psi\} \quad \{\varphi \land \neg e\} \ s_2 \ \{\psi\}}{\{\varphi\} \ \text{if $e$ then $s_1$ else $s_2$ fi } \{\psi\}} \quad \frac{}{\{\varphi\} \ \text{while $e$ do $s$ od}}$$

## Assignment Project Exam Help

```
(\sigma_1, s_1) \Downarrow \sigma_2 (\sigma_2, s_2) \Downarrow \sigma_3 \{\varphi\} s_1 \{\alpha\} \{\alpha\} s_2 \{\psi\}
                                                                                                                                                              h^{(\sigma_1, s_1)} h^{s_2} h^{\sigma_3} h^{\sigma_3}
```

# $\overbrace{Add}^{(\sigma,x:=e) \ \Downarrow \ (\sigma:x\mapsto v)} \overbrace{Add}^{\{\varphi[x:=e]\} \ x:=e \ \{\varphi\}}$

$$\frac{\{\varphi \land e\} \ s_1 \ \{\psi\} \quad \{\varphi \land \neg e\} \ s_2 \ \{\psi\}}{\{\varphi\} \ \text{if } e \ \text{then} \ s_1 \ \text{else} \ s_2 \ \text{fi} \ \{\psi\}} \qquad \frac{\{\varphi \land e\} \ s \ \{\varphi\}}{\{\varphi\} \ \text{while} \ e \ \text{do} \ s \ \text{od}}$$

## Assignment Project Exam Help

```
(\sigma_1, s_1) \Downarrow \sigma_2 (\sigma_2, s_2) \Downarrow \sigma_3 \{\varphi\} s_1 \{\alpha\} \{\alpha\} s_2 \{\psi\}
                                                                                                                                                              h^{(\sigma_1, s_1)} h^{s_2} h^{\sigma_3} h^{\sigma_3}
```

# $\overbrace{Add}^{(\sigma,x:=e) \ \Downarrow \ (\sigma:x\mapsto v)} \overbrace{Add}^{\{\varphi[x:=e]\} \ x:=e \ \{\varphi\}}$

$$\frac{\{\varphi \wedge e\} \ s_1 \ \{\psi\} \quad \{\varphi \wedge \neg e\} \ s_2 \ \{\psi\}}{\{\varphi\} \ \text{if $e$ then $s_1$ else $s_2$ fi } \{\psi\}} \qquad \frac{\{\varphi \wedge e\} \ s \ \{\varphi\}}{\{\varphi\} \ \text{while $e$ do $s$ od } \{\varphi \wedge \neg e\}}$$

#### Consequence

## Assignment Project Exam Help There is one more rule, called the rule of consequence, that we need to insert ordinary

There is one more rule, called the *rule of consequence*, that we need to insert ordinary logical reasoning into our Hoare logic proofs:

### Add WeChat powcoder

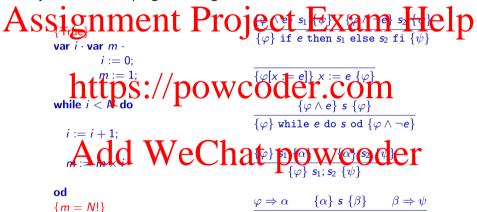
#### Consequence

## Assignment Project Exam Help There is one more rule, called the rule of consequence, that we need to insert ordinary

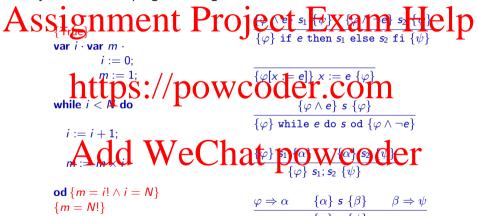
There is one more rule, called the *rule of consequence*, that we need to insert ordinary logical reasoning into our Hoare logic proofs:

This is the only rule that is not directed entirely by syntax. This means a Hoare logic proof need not tooklike a derivation tree librates when the consequence rule.

Let's verify the Factorial program using our Hoare rules:



Let's verify the Factorial program using our Hoare rules:



Let's verify the Factorial program using our Hoare rules:

#### Assignment Project si Folkam si Help var i · var m · i := 0: $\{ \underset{\text{while } i < A}{\text{hos}} : //powc \overset{\{\varphi[x] = e]\}}{\underset{\{\varphi \land e\}}{\text{od}}} : \overset{=}{\underset{s}{\text{e}}} : \varphi \}$ $\{\varphi\}$ while e do s od $\{\varphi \land \neg e\}$ Add WeChat poweder od $\{m = i! \land i = N\}$ $\beta \Rightarrow \psi$ $\{m = N!\}$

Let's verify the Factorial program using our Hoare rules:

```
Assignment Project si Folkam si Help
      var i · var m ·
           i := 0:
      \{\varphi\} while e do s od \{\varphi \land \neg e\}
       Add WeChat powcoder
       \{m = i!\}
      od \{m = i! \land i = N\}
                                              \beta \Rightarrow \psi
      \{m = N!\}
```

Let's verify the Factorial program using our Hoare rules:

#### Assignment Project si Folkam si Help var i · var m · i := 0: $\{ \underset{\text{while } i < N \text{ do } \{m = i\}}{\text{mattps:}} : // \underset{\text{power}}{\text{power}} \text{power} \text{oder } \overset{\{\varphi[x = e]\}}{\text{varies}} \xrightarrow{x := e} \overset{\{\varphi\}}{\{\varphi\}}$ $\{\varphi\}$ while e do s od $\{\varphi \land \neg e\}$ Add WeChat powcoder $\{m = i!\}$ od $\{m = i! \land i = N\}$ $\beta \Rightarrow \psi$ $\{m = N!\}$

Let's verify the Factorial program using our Hoare rules:

# Assignment Project $s_1$ Fix and $s_2$ Help var $p_1$ var $p_2$ var $p_3$ var $p_4$ var $p_4$ var $p_5$ var $p_6$ var $p_7$ var $p_8$ v

```
i := 0:
\{ \underset{\text{while } i < N \text{ do } \{m = i\}}{\text{mattps:}} : // \underset{\text{power}}{\text{power}} \text{power} \text{oder } \overset{\{\varphi[x = e]\}}{\text{varies}} \xrightarrow{x := e} \overset{\{\varphi\}}{\{\varphi\}} 
                                                                     \{\varphi\} while e do s od \{\varphi \land \neg e\}
    Add WeChat powcoder
     \{m = i!\}
od \{m = i! \land i = N\}
                                                                                                                     \beta \Rightarrow \psi
\{m = N!\}
```

Let's verify the Factorial program using our Hoare rules:

# Assignment Project si Fox am si Help

```
var i · var m ·
         i := 0:
  \{m \times (i+1) = (i+1)!\}
                                        \{\varphi\} while e do s od \{\varphi \land \neg e\}
  i := i + 1:
            d WeChat-poweoder
  \{m = i!\}
od \{m = i! \land i = N\}
                                                                    \beta \Rightarrow \psi
\{m = N!\}
```

Let's verify the Factorial program using our Hoare rules:

# Assignment Project si Fox am si Help

```
var i · var m ·
              i := 0:
\{ \underset{\text{while } i < N \text{ do } \{m = i}{\text{the } i < N \}} \text{ Code } \{ \underset{\{\varphi \land e\}}{\text{code } s \in \{\varphi\}} \} \underbrace{x := e }_{\{\varphi\}} \{ \varphi \}
    \{m \times (i+1) = (i+1)!\}
                                                              \{\varphi\} while e do s od \{\varphi \land \neg e\}
    i := i + 1:
                   d WeChat-poweoder
    \{m = i!\}
od \{m = i! \land i = N\}
                                                                                                        \beta \Rightarrow \psi
\{m = N!\}
```



Let's verify the Factorial program using our Hoare rules:

# Assignment Project si Fox am si Help

```
var i · var m ·
         i := 0:
  \{m \times (i+1) = (i+1)!\}
                                        \{\varphi\} while e do s od \{\varphi \land \neg e\}
  i := i + 1:
            d WeChat-poweoder
  \{m = i!\}
od \{m = i! \land i = N\}
                                                                    \beta \Rightarrow \psi
\{m = N!\}
```



Let's verify the Factorial program using our Hoare rules:

# Assignment Project $s_1$ For $s_2$ the left $s_2$ for $s_3$ the left $s_4$ for $s_4$ then $s_4$ else $s_2$ for $\{\psi\}$

```
var i · var m ·
         i := 0:
  \{m \times (i+1) = (i+1)!\}
                                        \{\varphi\} while e do s od \{\varphi \land \neg e\}
  i := i + 1:
            d WeChat-poweoder
  \{m = i!\}
od \{m = i! \land i = N\}
                                                                    \beta \Rightarrow \psi
\{m = N!\}
```



Let's verify the Factorial program using our Hoare rules:

# Assignment Project si Fox am si Help

```
var i · var m ·
         i := 0; \{1 = i!\}
  \{m \times (i+1) = (i+1)!\}
                                         \{\varphi\} while e do s od \{\varphi \land \neg e\}
  i := i + 1:
            d WeChat-poweoder
  \{m = i!\}
od \{m = i! \land i = N\}
                                                                     \beta \Rightarrow \psi
\{m = N!\}
```



Let's verify the Factorial program using our Hoare rules:

# Assignment Projects Exams Help

```
var i · var m ·
\{1 = 0!\} i := 0; \{1 = i!\}
  \{m \times (i+1) = (i+1)!\}
                                         \{\varphi\} while e do s od \{\varphi \land \neg e\}
  i := i + 1:
            d WeChat-poweoder
  \{m = i!\}
od \{m = i! \land i = N\}
                                                                     \beta \Rightarrow \psi
\{m = N!\}
```

