Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

CS 320 : Operational Semantics Scope

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Marco Gaboardi
MSC 116
gaboardi@bu.edu

Announcements

- ***Programming Assignment #4*** is due on Friday, April 3.
- ***Grading Policy for Spring 2020***: Read the article in **BU Today**, University to Offer Students Credit/No Credit Option
- ***Programming Assignment #5*** will be posted on Friday, April 3.
- ***TopHat questions*** are included in Zoom meetings, but not graded.
- All ***Zoom meetings*** are recorded (by default), and their recordings available for download shortly after the live meetings.
- All ***Zoom links*** for CS 320 are at the bottom of the *Resources* webpage on *Piazza*.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Learning Goals for today

- To understand how the dynamic semantics of a program describes the meaning of a program.
- To understand the subtleties of different evaluation strategies.
- Digging into the concepts of binding and scope.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Arithmetical expressions: shape of expressions

- Let us consider this simple language for expressions

```
<expr> ::= <expr> <addop> <term> | <term>
<addop>::= add | minus
<term> ::= var | val
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Arithmetical expressions, a little more general than the grammar in slides 23, 24, 26 of Lecture 17:

⟨expr⟩ ::= ⟨term⟩ | (⟨expr⟩ ⟨addop⟩ ⟨expr⟩)

⟨addop⟩ ::= add | minus

⟨term⟩ ::= ⟨var⟩ | ⟨val⟩

⟨var⟩ ::= ...

⟨val⟩ ::= ...

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Examples of arithmetical expressions generated by this BNF grammar:

(((3 add 5) minus 6) add (2 minus 1))

(((X add 5) minus Y) add (2 minus Z))

Operational semantics for arithmetical expressions

$$(e/m) \rightarrow (e/m)$$

Assignment Project Exam Help

Here (e/m) is a configuration where e is an expression and m is a memory.

https://powcoder.com

Add WeChat powcoder

We can think about a memory as a set of (unique) assignments of variables to values:

$$m = ((x_1=v_1), (x_2=v_2) \ldots, (x_n=v_n))$$

The memory is the environment where the variable of an expression are defined.

Arithmetical expressions: shape of expressions

- Let us consider this simple language for expressions

```
<expr> ::= <expr> <addop> <term> | <term>
<addop>::= add
<term> ::= var | val
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- What is the potential shape of an expression?
  - `v` — Value
  - `x` — Variable
  - `e add (n|x)` — Expression + Constant or Variable

This is recursive

Summary of the rules:

(F) $(x/m) \rightarrow (\mathtt{fetch}(x,m)/m)$

Assignment Project Exam Help

(+A) $(e \ \mathtt{add} \ x/m) \rightarrow (e \ \mathtt{add} \ \mathtt{fetch}(x,m)/m)$

https://powcoder.com

(+B) $(x \ \mathtt{add} \ v/m) \rightarrow (\mathtt{fetch}(x,m) \ \mathtt{add} \ v/m)$

Add WeChat powcoder

(+C) $(v_1 \ \mathtt{add} \ v_2/m) \rightarrow (v_1+v_2/m)$

(+D) $$\frac{(e/m) \rightarrow (e_1/m)}{(e \ \mathtt{add} \ v_2/m) \rightarrow (e_1 \ \mathtt{add} \ v_2/m)}$$

Summary of the rules:

(-A) $(e \text{ minus } x/m) \rightarrow (e \text{ minus } \text{fetch}(x,m)/m)$

Assignment Project Exam Help

(-B) $(x \text{ minus } v/m) \rightarrow (\text{fetch}(x,m) \text{ minus } v/m)$

https://powcoder.com

(-C) $(v_1 \text{ minus } v_2/m) \rightarrow (v_1 - v_2/m)$

Add WeChat powcoder

(-D) $$\frac{(e/m) \rightarrow (e_1/m)}{(e \text{ minus } v_2/m) \rightarrow (e_1 \text{ minus } v_2/m)}$$

Evaluation of (((3 add 5) minus 6) add (2 minus 1))
relative to the memory ( ).
We organize the evaluation differently from that in
Professor Gabbardi's slides:

(((3 add 5) minus 6) add (2 minus 1)) →

Assignment Project Exam Help

((8 minus 6) add (2 minus 1)) →

https://powcoder.com

(2 add (2 minus 1)) →

Add WeChat powcoder

(2 add 1) →

3

Evaluation of (((X add 5) minus Y) add (2 minus Z))
relative to the memory (X=3, Y=6, Z=1) = m ;
again we organize the evaluation differently from
that in Professor Gaboardi's slides:

(((X add 5) minus Y) add (2 minus Z)) / m →

Assignment Project Exam Help

(((3 add 5) minus Y) add (2 minus Z)) / m →

https://powcoder.com

(((3 add 5) minus 6) add (2 minus Z)) / m →

Add WeChat powcoder

(((3 add 5) minus 6) add (2 minus 1)) / m →

⋮

(2 add 1) / m →

3

Assignment Project Exam Help

TopHat Q1 - Q3

https://powcoder.com

Add WeChat powcoder

Assignment Project Exam Help

Variables

https://powcoder.com

Add WeChat powcoder

Variables

- Functional languages use variables as names (where the association name-value is stored in an environment).
  - We can remember the association, or read the value, but we cannot change it.
- Imperative languages are abstractions of von Neumann architecture
  - A variable abstracts the concept of memory location
- Understanding how variables are managed is an important part to understand the semantics of a programming language.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

Let in a Functional Language

https://powcoder.com

Add WeChat powcoder

Let expression

- Let us consider this simple language for expressions

```
<expr> ::= let var= <expr> in <expr> |
           <expr> <addop> <term> | <term>
<addop>::= add
<term> ::= var | val
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- What is the semantics of a let expression?

Another Example

```
let x = 2 in
let y = x + x in
y * x
```

substitute
2 for x

-->

```
let y = 2 + 2 in
y * 2
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Moral: Let operates by *substituting* computed values for variables

-->

```
let y = 4      in
y * 2
```

substitute
4 for y

-->

```
4 * 2
```

-->

```
8
```

Rules for let

$$\frac{(e_1/m) \rightarrow (e_3/m)}{(\texttt{let } x{=}e_1 \texttt{ in } e_2/m) \rightarrow (\texttt{let } x{=}e_3 \texttt{ in } e_2/m)}$$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

We can use the fact that e_1 is recursive and hypothetical reasoning.

Recording the value of a variable

$(\texttt{let } x=v \texttt{ in } e/m) \rightarrow ??(e/m@(x=v))$

Assignment Project Exam Help

If variables are just names for values, where shell we store the name-value association?

https://powcoder.com

This is the role of the environment, storing name-value associations.

Add WeChat powcoder

$$((x_1=v_1),(x_2=v_2)\ldots,(x_n=v_n))$$

Where we use the symbol @ to extend the environment with a new name-value association.

Extending an environment

What happens if `m` already contains `u`?

Suppose that we have

```
m=((x=1),(z=5),(y=3))
```

Assignment Project Exam Help

Then, if we extend m with the new pair (u=4), in symbols

https://powcoder.com

Add WeChat powcoder

```
m@(u=4)
```

We get:

```
m@(u=4)=((x=1),(z=5),(y=3),(u=4))
```

Summary of the rules:

(F) $(x/m) \rightarrow (\mathtt{fetch}(x,m)/m)$

(A) $(e \ \mathtt{add}\ x/m) \rightarrow (e \ \mathtt{add}\ \mathtt{fetch}(x,m)/m)$

(B) $(x \ \mathtt{add}\ v/m) \rightarrow (\mathtt{fetch}(x,m) \ \mathtt{add}\ v/m)$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

(C) $(v_1 \ \mathtt{add}\ v_2/m) \rightarrow (v_1+v_2/m)$

(D) $$\frac{(e/m) \rightarrow (e_1/m)}{(e \ \mathtt{add}\ v_2/m) \rightarrow (e_1 \ \mathtt{add}\ v_2/m)}$$

(T) $$\frac{(e_1/m) \rightarrow (e_3/m)}{(\mathtt{let}\ x=e_1\ \mathtt{in}\ e_2/m) \rightarrow (\mathtt{let}\ x=e_3\ \mathtt{in}\ e_2/m)}$$

(L) $(\mathtt{let}\ x=v\ \mathtt{in}\ e/m) \rightarrow (e/m@(x=v))$

Example:

Let us call m= (x=3,y=5,z=6) we have:

(x add y/m) → (x add 5/m)

(let k=(x add y)in(k add z)/m) →(let k=(x add 5)in(k add z)/m) =

(x add 5/m) → (3 add 5/m)

(let k=(x add 5)in(k add z)/m)→(let k=(3 add 5)in(k add z)/m) =

(3 add 5/m) → (8/m)

(let k=(3 add 5)in(k add z)/m)→(let k=8 in(k add z)/m) →

(k add z/m@(k=8))→(k add 6/m@(k=8))→(8 add 6/m@(k=8))

→(14/m@(k=8))

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

TopHat Q4 - Q6

https://powcoder.com

Add WeChat powcoder

Assignment Project Exam Help

Update in a Imperative Language

https://powcoder.com

Add WeChat powcoder

Assignment

- Let us consider this simple language for expressions

```
<prog>   ::= <assgn> | <assgn> ; <prog>
<assgn>  ::= var := <expr>
<expr>   ::= <expr> <addop> <term> | <term>
<addop>  ::= add
<term>   ::= var | val
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- What is the semantics of an assignment?

Operational semantics for programs with assignments

$$(prog/m) \rightarrow (prog/m)$$

Assignment Project Exam Help

Here $(prog/m)$ is a configuration where prog is a program and m is a memory.

https://powcoder.com

Add WeChat powcoder

We can think about a memory as a set of (unique) assignments of variables to values:

$$m = ((x_1=v_1), (x_2=v_2) \ldots, (x_n=v_n))$$

The memory is the environment where the variable of an expression are defined.

Rules for assignment

Assignment Project Exam Help

$$\frac{(e_1/m) \rightarrow (e_2/m)}{(x:=e_1;prog/m) \rightarrow (x:=e_2;prog/m)}$$

https://powcoder.com
Add WeChat powcoder

We can use thypothetical reasoning.

An example: recording the value of a variable

`(x:=v;prog/m)` → `(prog/update(x,v,m))`

Assignment Project Exam Help

https://powcoder.com

Where we use the function `update(x,v,m)` to update the value of the variable `x` in `m` to `v`.

Add WeChat powcoder

Updating an environment

What happens if `m` does not contain `x`?

Suppose that we have

```
m=((x=1),(z=5),(y=3))
```

Assignment Project Exam Help

Then, if we update m with the following command

https://powcoder.com

Add WeChat powcoder

```
update(x,4,m)
```

We get:

```
update(x,4,m)=((x=4),(z=5),(y=3))
```

Initializing a variable

Suppose that we have

```
m=((u=1),(z=5),(y=3))
```

What shall we do if we have the following?

```
update(x,4,m)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Basically there are two strategies:
1- we create a new pair:

```
update(x,4,m)=((u=1),(z=5),(y=3),(x=4))
```

2- we give an error because the variable has not been initialized – how can we fix this?

Example:

Let us call m= (x=3,y=5,z=6,u=0) we have:

(x add 5/m) → (3 add 5/m)

(z:=x add 5 ; u:=u add z;p/m) → (z:=3 add 5 ; u:=u add z;p/m) =

(3 add 5/m) → (8/m)

(z:=3 add 5 ; u:=u add z;p/m) → (z:=8; u:=u add z;p/m) =

m'=(x=3,y=5,z=8,u=0)

(z:=8; u:=u add z;p/m) → (u:=u add z;p /m') =

(u add z/m') → (u add 8/m')

(u:=u add z;p/m') → (u:=u add 8;p/m') =

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Example:

(u add 8/m′) → (0 add 8/m′)

(u:=u add 8;p/m′)→(u:=0 add 8;p/m′) =

Assignment Project Exam Help

(0 add 8/m′) → (8/m′)

https://powcoder.com

(u:=0 add 8;p/m′)→(u:=8;p/m′) =

Add WeChat powcoder

m′′=(x=3,y=5,z=8,u=8)

(u:=8;p/m)→(p/m′′) → …

Assignment Project Exam Help

TopHat Q7 - Q9

https://powcoder.com

Add WeChat powcoder

Mutable vs Immutable Variables

- When we consider variables as names we are working with immutable variables (e.g. the part of OCaml we studied)
- When we consider variables as memory locations we are working with mutable variables (e.g. Python, c, etc.)
- Understanding how variables are managed is an important part to understand the semantics of a programming language.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mapping the formal semantics to an implementation

- Our formal rules for the interpreter language operated over "configurations" that contained the program (list of commands) and a stack (list of values with type tags)
- Consequence: lets write a function that operates over a list of commands and a stack

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
let rec foo commandList stack =
    match (commandList, stack) with
    | (Add::cs, I(x)::I(y)::s) -> foo cs I(x+y)::s
```

$(p/S) \rightarrow (p'/S')$

(B) $(add;p/int(v_2)::int(v_1)::S) \rightarrow (p/int(v_2+v_1)::S)$

Mapping the formal semantics to an implementation

- Our formal rules for the interpreter language operated over "configurations" that contained the program (list of commands) and a stack (list of values with type tags)
- Consequence: lets write a function that operates over a list of commands and a stack

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
let rec foo commandList stack =
    match (commandList, stack) with
    | (Add::cs, I(x)::I(y)::s) -> foo cs I(x+y)::s
    | (Div::cs, I(x)::I(0)::s) -> foo cs E::I(x)::I(0)::s
    | (Div::cs, I(x)::I(y)::s) -> foo cs I(x/y)::s
```

$$(\text{div};p/\text{int}(v_2)::\text{int}(0)::S) \rightarrow (p/(\text{:error:})::\text{int}(v_2)::\text{int}(0)::S)$$

$$(\text{div};p/\text{int}(v_2)::\text{int}(v_1)::S) \rightarrow (p/\text{int}(v_2/v_1)::S) \quad v_1 \neq 0$$

What about this?

Mapping the formal semantics to an implementation

- Our formal rules for the interpreter language operated over "configurations" that contained the program (list of commands) and a stack (list of values with type tags)
- Consequence: lets write a function that operates over a list of commands and a stack

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Can also write it this ways

```
let rec foo commandList stack =
    match (commandList, stack) with
    | (Add::cs, I(x)::I(y)::s) -> foo cs I(x+y)::s
    | (Div::cs, I(x)::I(0)::s) -> foo cs E::stack
    | (Div::cs, I(x)::I(y)::s) -> foo cs I(x/y)::s
```

$$\frac{}{(\mathtt{div;p/int}(v_2)\mathtt{::int(0)::S}) \rightarrow (\mathtt{p/(:error:)::int}(v_2)\mathtt{::int(0)::S})}$$

$$\frac{v_1 \neq 0}{(\mathtt{div;p/int}(v_2)\mathtt{::int}(v_1)\mathtt{::S}) \rightarrow (\mathtt{p/int}(v_2/v_1)\mathtt{::S})}$$

Operational semantics for the interpreter

$$(p/S) \rightarrow (p'/S')$$

Assignment Project Exam Help

Here (p/S) is a configuration where p is a program and S is a stack. We call these pairs configurations because we think in terms of an "abstract machine".

https://powcoder.com

Add WeChat powcoder

We can think about the stack as a list of values (denoted with v):

$$v_n :: \ldots :: v_2 :: v_1 :: []$$

We say that from the configuration (p/S) we can step (or reduce) to the configuration (p'/S') in one step.

Operational semantics for arithmetical expressions

$$(e/m) \rightarrow (e/m)$$

Assignment Project Exam Help

Here (e/m) is a configuration where e is an expression and m is an environment. We call these pairs configurations because we think in terms of an "abstract machine".

https://powcoder.com

Add WeChat powcoder

We can think about an environment as a set of (unique) assignments of variables to values:

$$m = ((x_1=v_1), (x_2=v_2) \ldots, (x_n=v_n))$$

Tips for interpreter part2:
Operational semantics for the interpreter with variables

$$(p/S,m) \rightarrow (p'/S',m')$$

Assignment Project Exam Help

Here $(p/S,m)$ is a configuration where p is a program and S is a stack, and m is an environment.

https://powcoder.com

Add WeChat powcoder

We can think about the stack as a list of values:

$$v_n::\ldots::v_2::v_1::[]$$

We can think about an environment as a set of (unique) assignments of variables to values:

$$m = ((x1=v1),(x2=v2)\ldots,(xn=vn))$$

Tip for interpreter part2: Implementation of OCaml `let`

We could imagine the let construction we saw in OCaml and in the last class:

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
let x=v in ...
```

to be implemented as

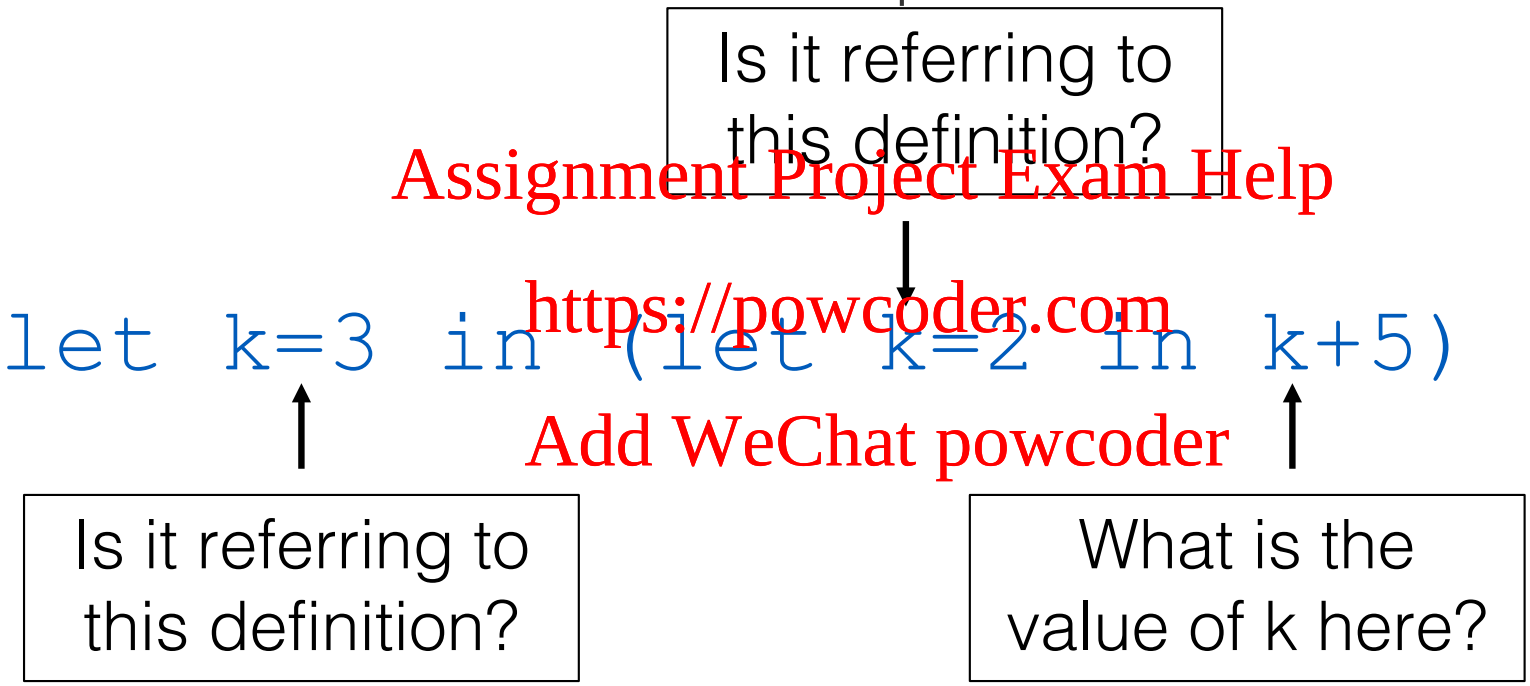
```
pushI v
pushN x
  bind
```

...

Assignment Project Exam Help
https://powcoder.com
Scoping rules
Add WeChat powcoder

Variable names

How shall we evaluate this expression?



Scope of a variable

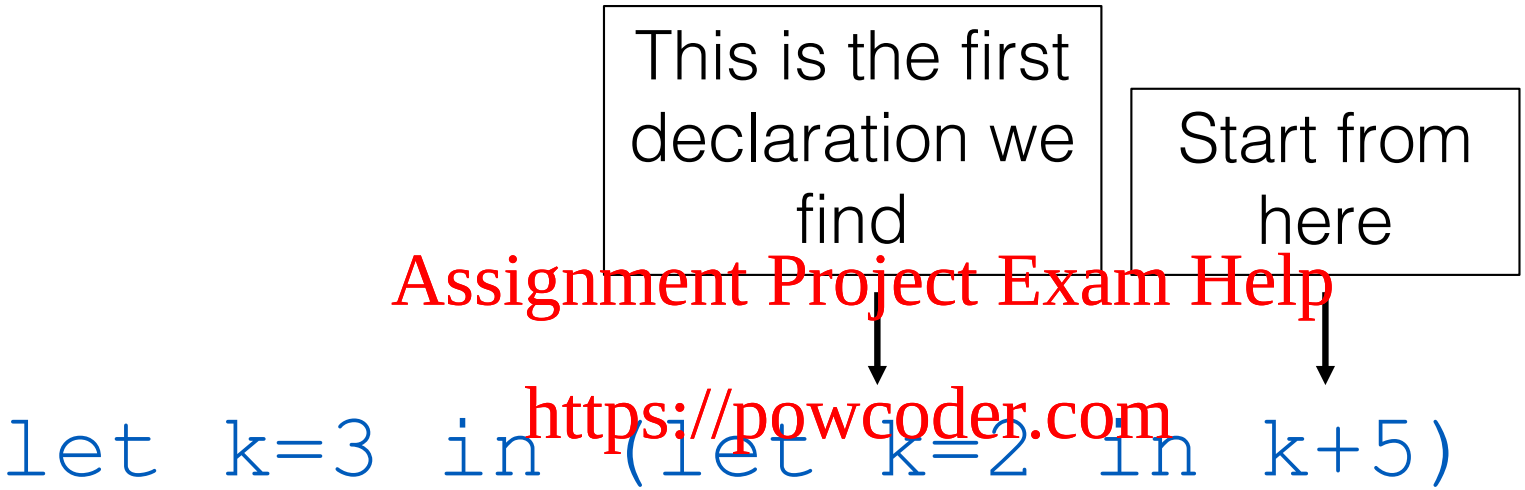
- The scope of a variable is the range of statements over which it is visible
- The scope rules of a language determine how references to names are associated with variables

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
let k=3 in (let k=2 in k+5)
```

OCaml scoping rule says that a variable name is statically associated with the closest definition in the abstract syntax tree.

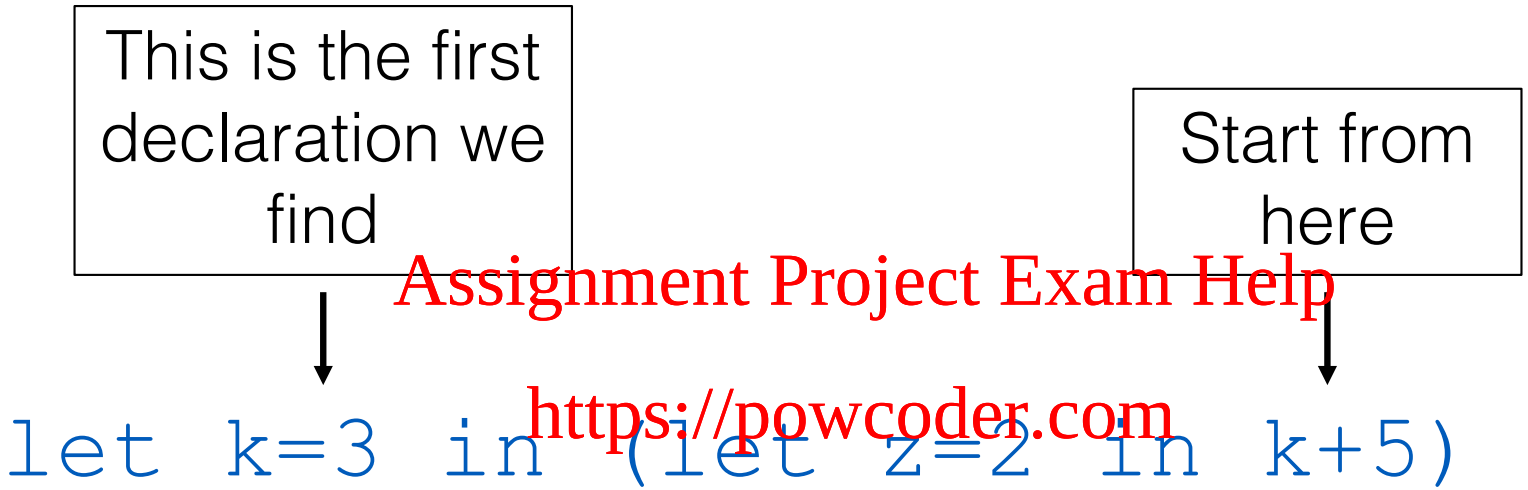
Back to our example



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

To find the value of k we look search declarations, first locally, then in increasingly larger enclosing scopes

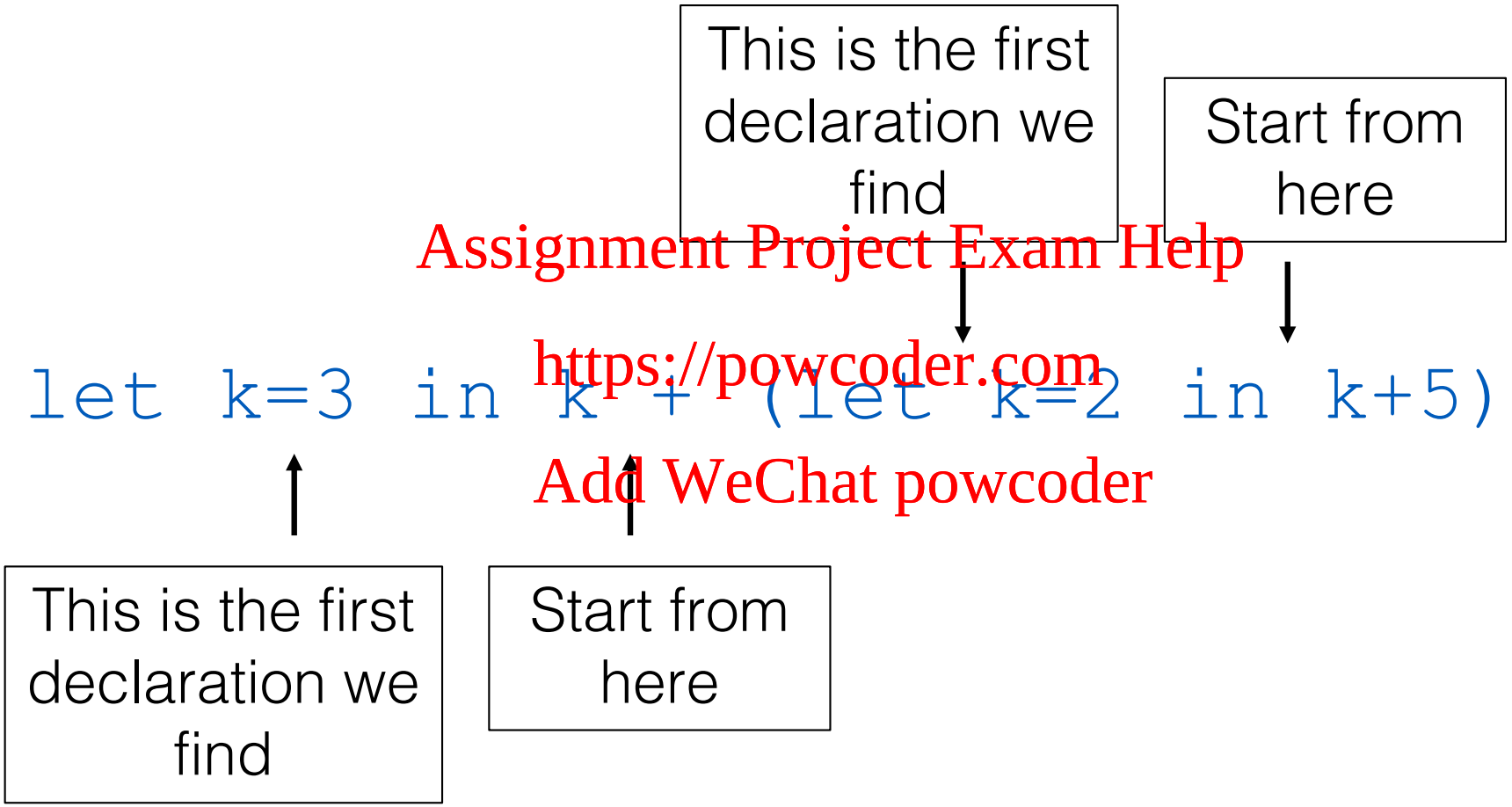
Another example



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

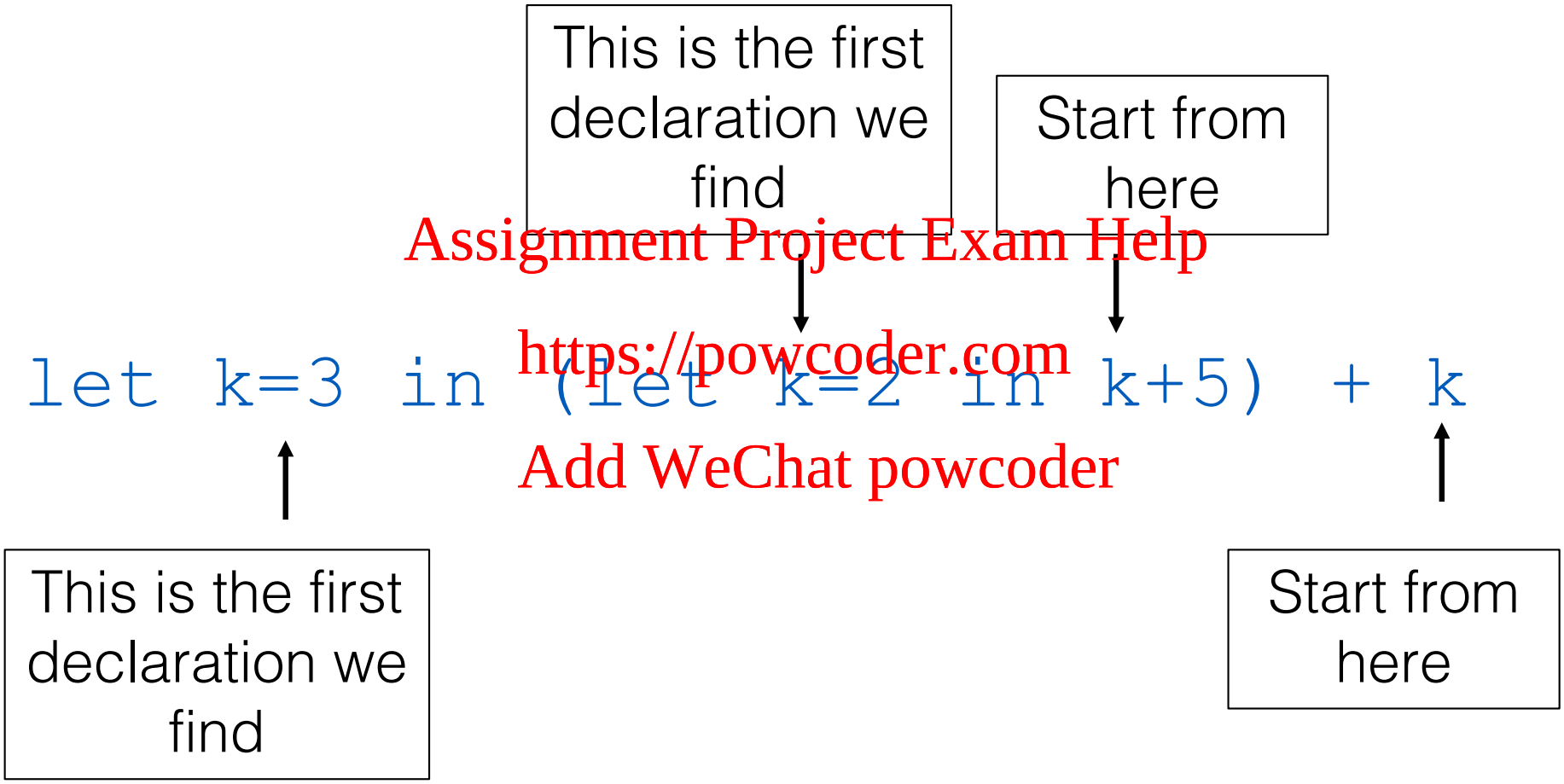
To find the value of k we look search declarations, first locally, then in increasingly larger enclosing scopes

Another example



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Another example



Static Scope

- Based on program text
- To connect a name reference to a variable, we (or the compiler) must find the declaration
- Some languages allow nested subprogram definitions, which create nested static scopes
- Search process:
  search declarations, first locally, then in increasingly larger enclosing scopes, until one is found for the given name

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

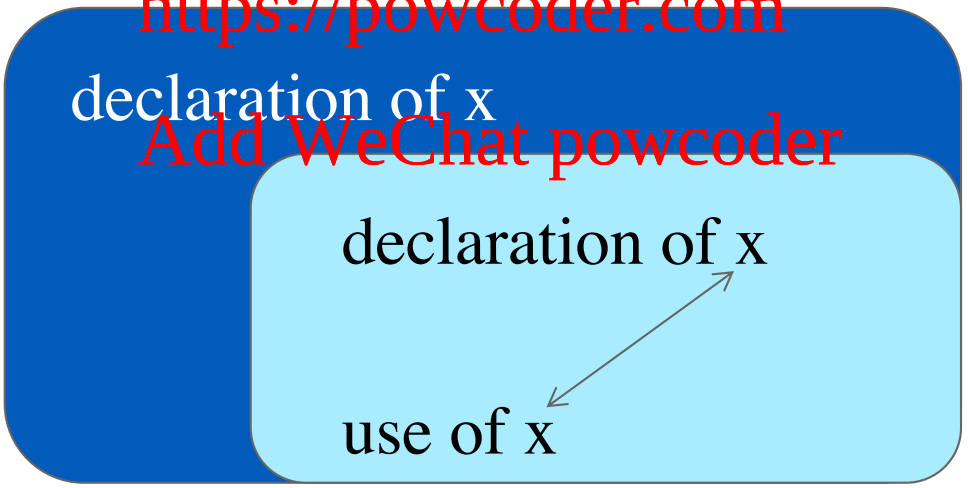
Static Scope

- Variables can be hidden from a unit by having a "closer" variable with the same name

Assignment Project Exam Help

https://powcoder.com

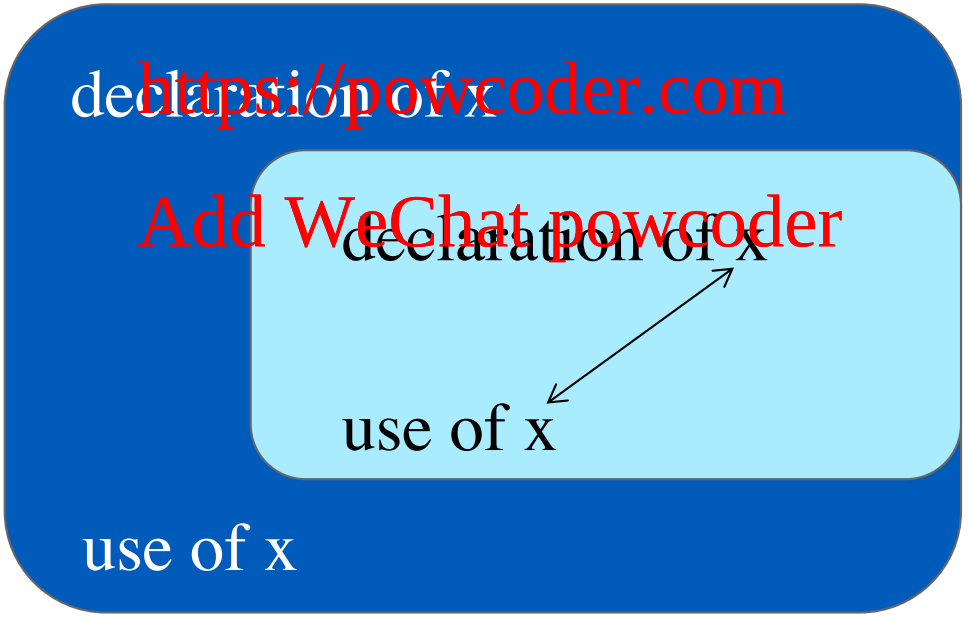
Add WeChat powcoder



Static Scope

- Variables can be hidden from a unit by having a "closer" variable with the same name

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Static Scope

- Search process:
  search declarations, first locally, then in increasingly larger enclosing scopes, until one is found for the given name
- Enclosing static scopes (to a specific scope) are called its static ancestors; the nearest static ancestor is called a static parent

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

TopHat Q10 - Q12

https://powcoder.com

Add WeChat powcoder

Scope Blocks

A method of creating static scopes inside program units (ALGOL 60)

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
void sub() {
   int count;
   while (...) {
    int count;
        count++;
         ...
     }
   …
}
```

Program constructs (“blocks”) create scopes

Scope Block Example:

```
int main()
{
   int x=5;
   {
      int x=4;
      printf("The value of x in the block is %d\n", x);
   }
   printf("The value of x in outside the block is %d",
x);
   return 0;
}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

In C we can write a program like the one above

Scope Block Example:

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
int main()
{
   int x=5;
   {
      int x=4;
      printf("The value of x in the block is %d\n", x);
   }
   printf("The value of x in outside the block is %d",
x);
   return 0;
}
```

```
main.c: In function ‘main’:
main.c:17:57: error: ‘x’ undeclared (first use in this function)
     printf("The value of x in outside the block is %d", x);
                                                         ^
```

Tip for interpreter part2: `let...end` construction

In the interpreter description for part 2 we require to implement a construction

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

`let ... end`

this is like a scope block `{ ... }`.