

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



#1

CMPSC 461: Programming Language Concepts

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

Instructor: Danfeng Zhang
W369 Westgate Building

TAs and LAs

Teaching Assistants:

- Zeyu Ding (Head TA, dxd437@psu.edu)
- Ashish Kumar (azk640@psu.edu)
- Madhav Deshpande (mzd574@psu.edu)
- Namitha Nambiar (nmn5265@psu.edu)

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Learning Assistants:

- Liang Leo (hql5432@psu.edu)
- Jianyu He (jvh6056@psu.edu)

Office hours will be announced before next week

Course Mode: Remote Synchronize

<https://psu.zoom.us/j/91404124375?pwd=WWRD SG16SX R1VFJUMy9qTHE3dmtmQT09>

Attendance Assignment Project Exam Help

- Regular attendance is ~~strongly recommended~~
- However, to accommodate special needs:
 - All lectures will be recorded and uploaded to Canvas
 - Physical attendance will *not* be used as part of course assessment

Important Learning Resources

- Course Homepage

<http://www.cse.psu.edu/~dbz5017/cmpsc461fa20>

Overview Schedule Piazza

Assignment Project Exam Help

All dates for lectures and unreleased assignments are provisional.

Lectures meet:

Section 1: 12:20pm–1:10pm MWF via ZOOM .

Section 2: 1:25pm–2:15pm MWF via ZOOM .

Required Book

(PLP) Programming Language Pragmatics, by Michael S. Tiernan (2018)

Add WeChat powcoder

Tentative Course schedule and reading materials

Lecture	Date	Topic/notes	Readings	Assignments, etc.
1	Aug 24	Course overview	PLP Sec 1	
		Functional Programming, Scheme Introduction		
2	Aug 26	Lambda Calculus	Note 1 on Canvas	
3	28	Binding	Note 1 on Canvas	
4	31	Reduction	Note 1 on Canvas	

Important Learning Resources

- Canvas

Home

Announcements

Assignments

Grades

People

Files

Syllabus

Modules

Zoom

Piazza

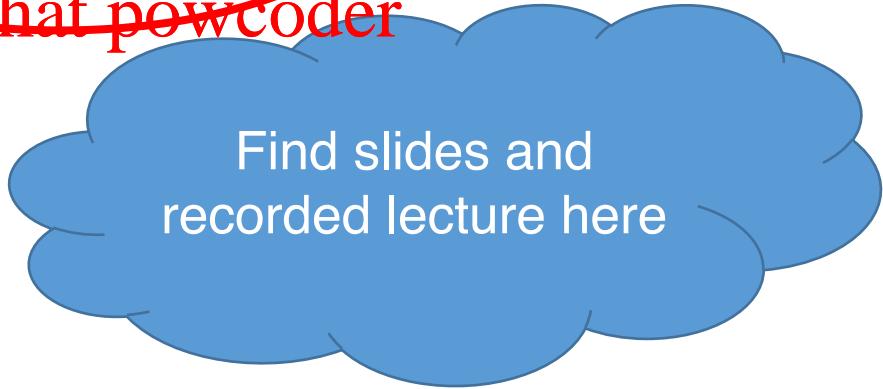
COVID Schedule

~~Assignment Project Exam Help~~

~~Aug 24, Course Introduction~~

~~https://powcoder.com~~

~~Add WeChat powcoder~~


Find slides and
recorded lecture here

Important Learning Resources

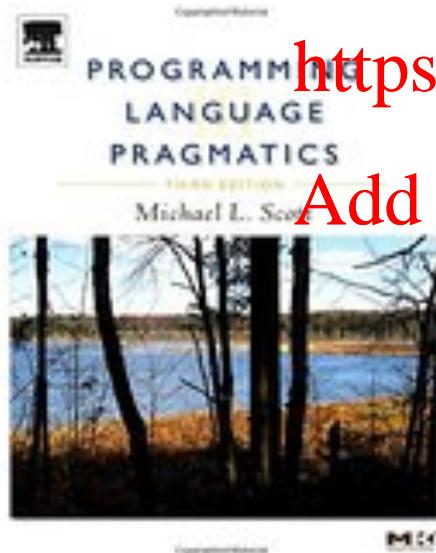
- Office hours (via Zoom) throughout the week
 - To be announced before next week
- Assignment Project Exam Help
- Piazza <https://powcoder.com>
<https://www.piazza.com/psu/fall2020/cmpsc461>
Add WeChat powcoder

Use Piazza for ALL course communication. Emails will typically be filtered/ignored.

Important Learning Resources

- Required Textbook

Programming Language Pragmatics
Assignment Project Exam Help

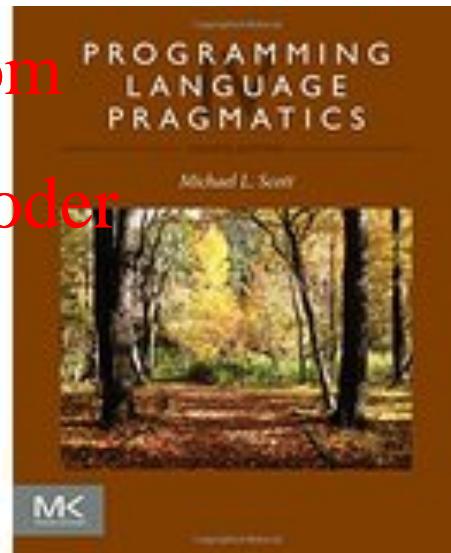


3rd Edition

<https://powcoder.com>

Add WeChat powcoder

OR



4th Edition

Administration

Assignments

- Written and programming
- No final projects

Assignment Project Exam Help

<https://powcoder.com>

Late policy

- 1 day late: -20%
- 2 days late: -50%
- >2 days late: -100%

Add WeChat powcoder

Exams: 2 midterms and 1 final (TBD)

Administration

Grading policy

- 30% assignments
- 40% midterm exams
- 30% final exam

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Letter grades

- A: [93-100] A-: [90-93) B+: [87-90) B: [83-87)
B-: [80-83) C+: [77-80) C: [70-77) D: [60-70)
F: [0-60)

Honors Option

- Schreyer scholars are welcome to take the honors option of this course
 - Extra reading materials
 - More challenging assignments
<https://powcoder.com>
- Send me emails to sign up for the honors option
[Add WeChat powcoder](#)

Academic Integrity

- Any form of sharing, plagiarism, copying, cheating will receive academic sanctions
~~Assignment Project Exam Help~~
- Any submitted work should be work of your own
- We will follow the [guideline of CSE department:](https://powcoder.com)
 - 0 for the submission that violates AI, AND
~~Add WeChat powcoder~~
 - a reduction of one letter grade for the final course grade
(students with prior AI violation will receive an F as the final course grade)

Assignment Project Exam Help

<https://powcoder.com>

Questions?
Add WeChat powcoder

Assignment Project Exam Help
Overview of CMPSC 461
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

CMPSC 461 *IS NOT*

- C/C++/Java/Python/Scheme/... programming
- Compiler construction (471)
- Object-oriented programming (221)
- Data structures

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461

Explores ***fundamental principles and paradigms*** of programming languages.

Assignment Project Exam Help

Studies features found in many different languages and examine how they work and how they interact with each other.

Why CMPSC 461?

Programming languages facilitate
communication of ideas

Assignment Project Exam Help

- Between people
- Between people and computers

<https://powcoder.com>

Add WeChat powcoder

This course explores the **fundamental principles** of programming languages, to facilitate communication in your future study/career

Why CMPSC 461?

Programming languages is a powerful tool once you master the principles

Assignment Project Exam Help

Example: use ~~type systems~~ to build secure software

Add WeChat powcoder

```
Int i =0;  
boolean b=true;  
b = i; // incompatible types: int  
      // cannot be converted to boolean
```

Why CMPSC 461?

Example: use **type systems** to build secure software

Assignment Project Exam Help

```
Secret s=0;  
Public p=1;  
p = s; //Add WeChat powcoder  
      // secret value to public variable
```

Designing new languages for building secure software is an active research area

Ways to Fail

- Ignore assignments
- Never attend lectures & office hours
Assignment Project Exam Help
- Skim through slides
https://powcoder.com
- Memorize assignments and practice questions
before exams
Add WeChat powcoder
- Miss exams (and their make-ups)

Ways to get an A? avoid all of the above!

Course Coverage

Programming
Paradigms

Imperative

Functional

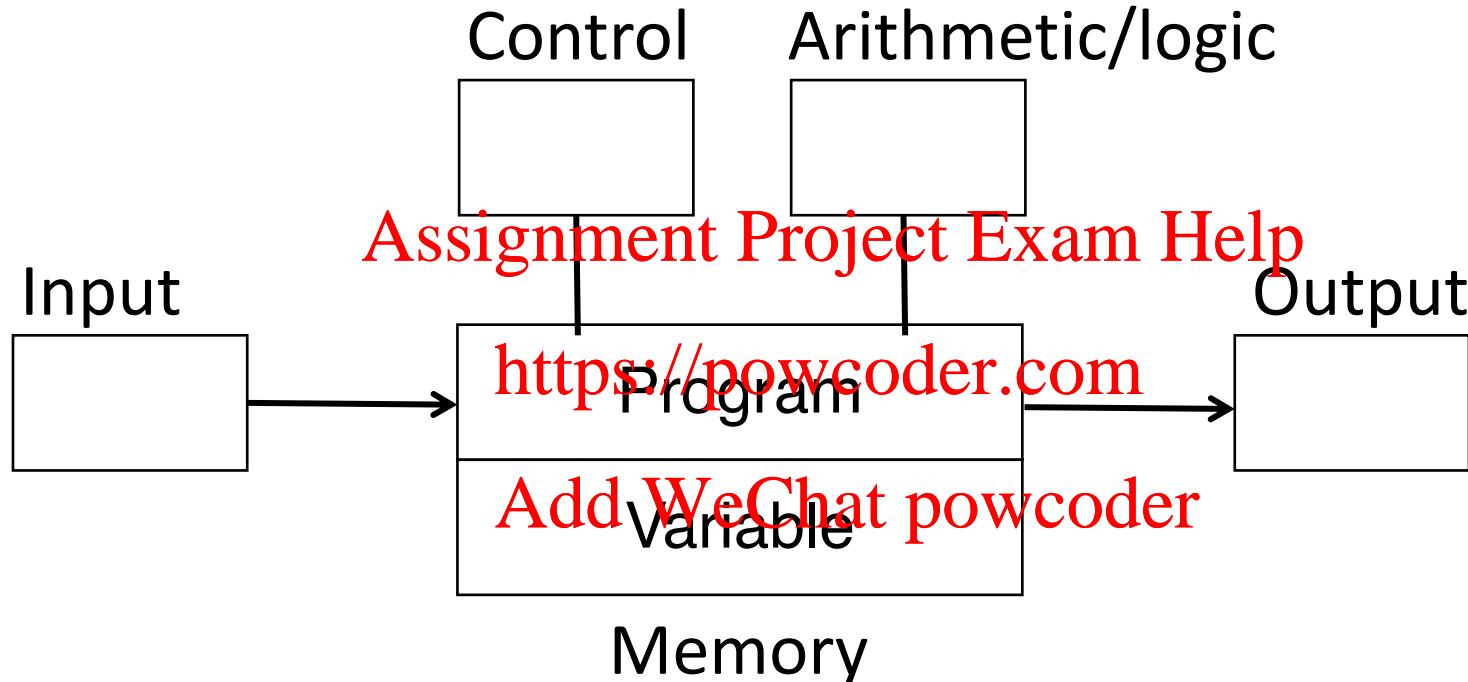
OOP

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Imperative Language



The von Neumann-Eckert Model, 1945

Imperative: program is updates to variables

Functional Language

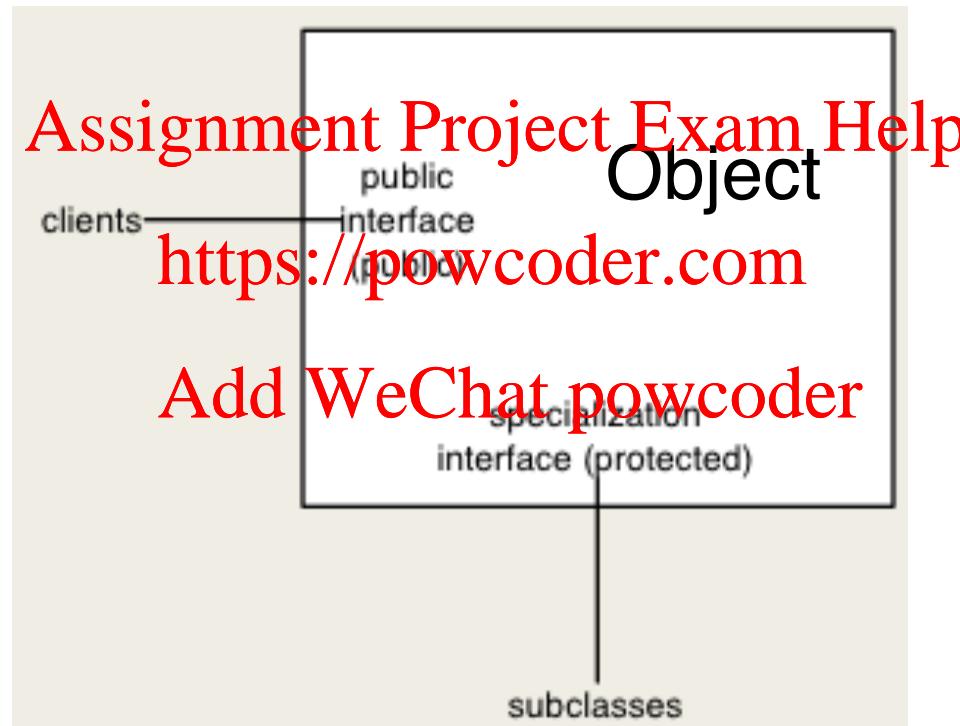
Assignment Project Exam Help

Output = f (Input)
<https://powcoder.com>

Add WeChat powcoder

Functional: program is a mathematical function

Object-Oriented Programming



Course Coverage

Programming
Paradigms

Imperative

Functional

OOP

Programming
Principals

Assignment Project Exam Help

Syntax

Names

Types

Semantics

Add WeChat powcoder

Correctness

Memory
Management

Language
Implementation



Levels



Focus of this
course

Higher-order language

Machine independent, e.g., $y = x+1$

Assignment Project Exam Help

Compiler

<https://powcoder.com>

Assembly language

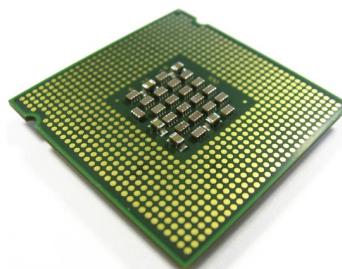
Processor instructions, e.g., MOV EAX, 1

Add WeChat powcoder

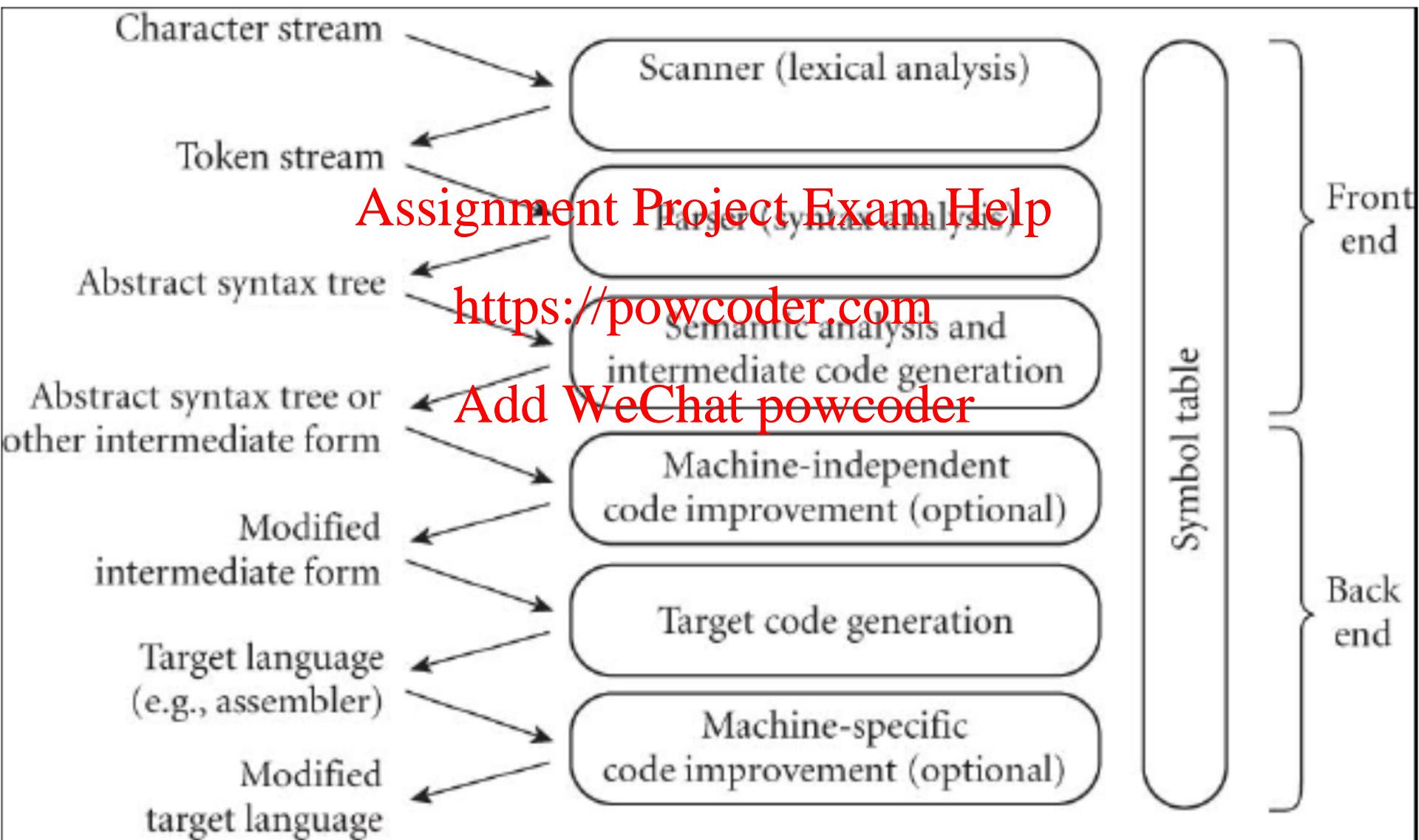
Assembler

Machine language

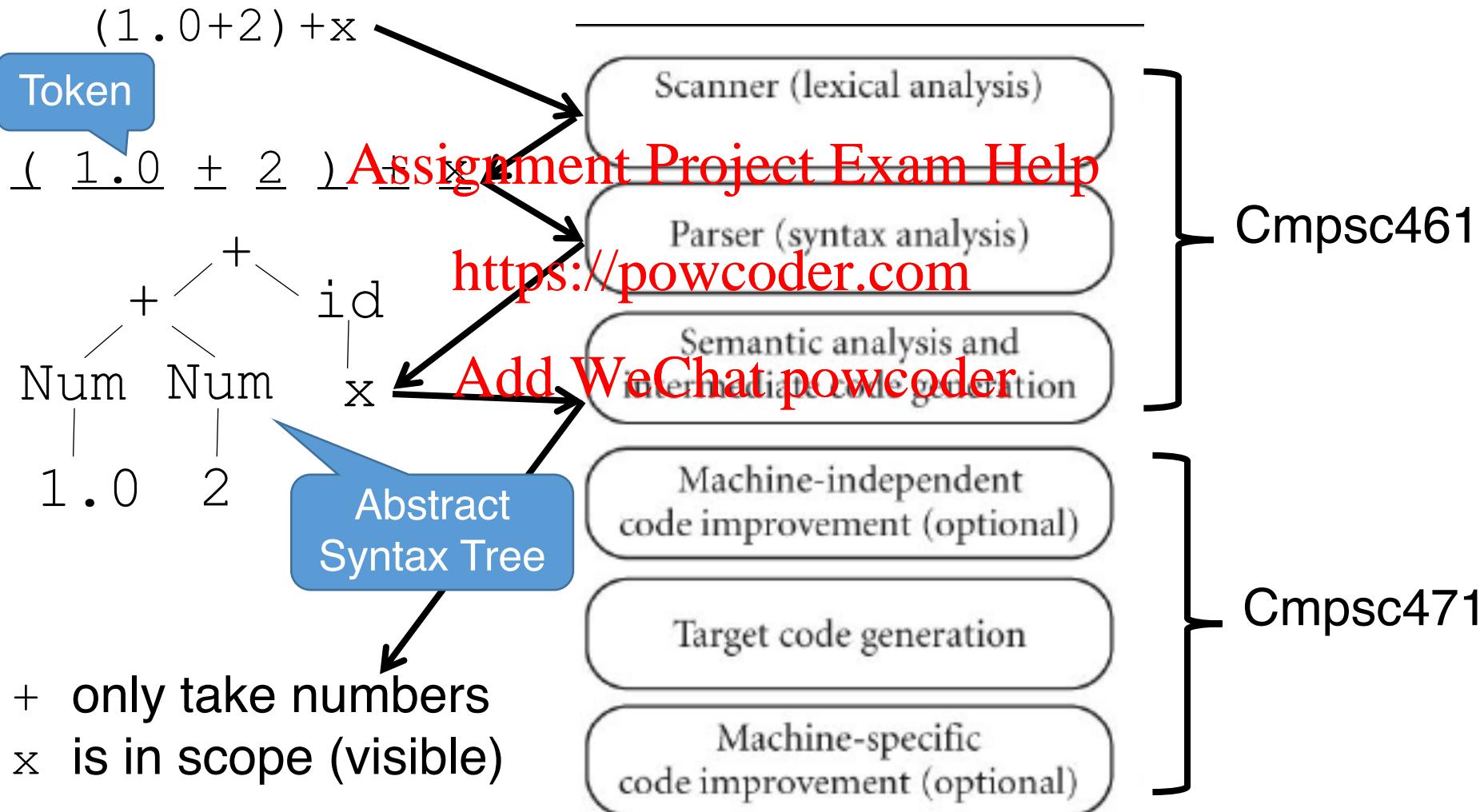
Patterns of bits, e.g., 00000101



Source Code to Target Code



Source Code to Target Code



Register the Piazza discussion group
<https://www.piazza.com/psu/fall2020/cmpsc461>
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

#2

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Functional Language

Assignment Project Exam Help

Output = f (Input)
<https://powcoder.com>

Add WeChat powcoder

Functional: program is a mathematical function

History

What is computation?

- Combinatory logic (1920's)
- *Lambda calculus* (1930's)
- Turing machine (1930's)
- ...

Add WeChat powcoder

First electronic general-purpose computer

- ENIAC (1946)

Mathematical Function

A mathematical function is a mapping of members of one set, called the **domain set**, to another set, called the **range set (image)**.

Notations:

<https://powcoder.com>

Domain

Range

Add WeChat powcoder
 $+ : (\mathbb{Z}, \mathbb{Z}) \mapsto \mathbb{Z}$

isNegative: $\mathbb{Z} \mapsto \{\text{True, False}\}$

Infix form

Function application

$+ 1 2 = 3$ (same as $1+2=3$)

isNegative 1 = False

The λ -Calculus

Alonzo Church, 1930s



Assignment Project Exam Help

A pure λ -term is defined inductively as follows:

- Any variable x is a λ -term
 - If t is a λ -term, so is $\lambda x. t$ (abstraction)
 - If t_1, t_2 are λ -terms, so is $t_1 t_2$ (application)
- $\lambda x. t$ λ Parameter
 $t_1 t_2$ separator like comma
apply function t_1 to t_2 return value
 $t ::= x / \lambda x. t / t_1 \cup t_2$

Analogy in C:

Abstraction: int f

Application: f(2)

λx t
(int x) . {return x+1}

λ -Term Examples

Identity function: $\lambda x. x$

Application: $(\lambda x. x) y$ [apply identity function to parameter y]

<https://powcoder.com>

How to parse a λ -term

Add WeChat powcoder

1. λ binding extends to the rightmost part

$\underline{\lambda x. (x (\lambda y. (y z)))}$ is parsed as $\lambda x. (x (\lambda y. (y z)))$

2. Applications are left-associative

$t_1 t_2 t_3$

is parsed as $(t_1 t_2) t_3$

value 1

$\underline{\lambda x. x \cup y}$

① $(\lambda x. x) \cup y$
 t_1 t_2

② $\lambda x. (x \cup y)$
 t_1

$x \cup y \cup z$

① $(x \cup y) \cup z$
 t_1 t_2

② $x \cup (y \cup z)$
 t_1 t_2

λ -Term Examples

Identity function: $\lambda x. x$

Application: $(\lambda x. x) y$ [apply identity function to
Assignment Project Exam Help parameter y]

<https://powcoder.com>

How to parse a λ -term

Add WeChat powcoder

1. λ binding extends to the rightmost part
 $\lambda x. x \lambda y. y z$ is parsed as $\lambda x. (x (\lambda y. (y z)))$
2. Applications are left-associative

$t_1 t_2 t_3$ is parsed as $(t_1 t_2) t_3$

Examples

$\lambda x.x (\lambda y.y) z$

① $\lambda x.(x(\lambda y.y) z$

↑
can't have another one inside (optional)

② $\lambda x.((x(\lambda y.y))z)$

t₁ t₂ t₃

x ↳ $\lambda x.y \quad \lambda x.z \quad \lambda z.y$

① $\lambda x.((y.x)t_1t_2(t_3))$ /powcoder.com

Add WeChat powcoder

Number of Parameters

In the pure λ -calculus, λ only bind one parameter

Assignment Project Exam Help
For convenience, we write

$\lambda x\ y.\ t$ as a ~~shorthand for $\lambda x.(\lambda y.\ t)$~~

Add WeChat powcoder

This process of removing parameters is called **currying**,
which we will cover later in this course.

#3

Functional Programming and Scheme

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Office hours

<https://psu.zoom.us/j/93599703460?pwd=TXEycTc2VmNRelZrb1pOYTh0YVRVUT09>

Assignment Project Exam Help

	Mon	Tue	Wed	Thu	Fri
8:00-9:00	Jianyu	Namitha	Madhav		
9:00-10:00	Jianyu	Namitha	Madhav		Leo
10:00-11:00	Ashish				
11:00-12:00	Ashish		Leo		
1:00-3:00				Zeyu	
2:30-3:30			Prof. Zhang		

The λ -Calculus

A pure λ -term t is defined inductively as follows:

- Any variable x is a λ -term
- If t is a λ -term, so is $\lambda x. t$ (abstraction)
- If t_1, t_2 are λ -terms, so is $t_1 t_2$ (application)

We use x, y, z, \dots for variables

The definition above defines an infinite set, named t

Syntax vs. Semantics

Syntax: the structure/form of lambda terms

$$t ::= x \mid t_0 t_1 \mid \lambda x. t$$

<https://powcoder.com>

Semantics: the meaning of lambda terms

- Lambda calculus defines computation
- What computation is defined by t ?

Capture-Avoiding Substitution

In λ -calculus, computation is defined by
capture-avoiding substitution

Assignment Project Exam Help

$t\{y/x\}$ means substitute *all free* x in t with y

replace all x in t with y

Add WeChat powcoder
 $(\lambda x. x) x \xrightarrow{\text{same } x?} (\lambda x. x) y$

bound

free

Analogy in C:

int x;

...

int f (int x) { return x }

Bound vs. Free Variables

In $(\lambda x. t)$, the variable x in t is **bound** to λx

A variable is **free** if it is not bound to any λ

A variable is ~~Assignment Project Exam Help~~

<https://powcoder.com>

Examples

$(\lambda x. x) x$ applies the identity function to x (i.e., the x after dot is bound to λ)

$\lambda x. \lambda x. x$ is a function that takes a parameter, and returns the identity function (i.e., the inner-most x is bound to the second λ) ~~bound to two λ~~

More Formally ...

In $(\lambda x. t)$, all free variables x in t is **bound** to λx

A variable is **free** if it is not bound to any λ

Assignment Project Exam Help

Systematically, we define free variables as follows:
<https://powcoder.com>

- $\text{FV}(x) = \{x\}$
- $\text{FV}(t_1 t_2) = \text{FV}(t_1) \cup \text{FV}(t_2)$
- $\text{FV}(\lambda x. t) = \text{FV}(t) - \{x\}$

Let $\text{Var}(t)$ be all variables used in t , the bound variables in t (written $\text{BD}(t)$) are

$$\text{BD}(t) = \text{Var}(t) - \text{FV}(t)$$

Examples

Systematically, we define free variables as follows:

- $\text{FV}(x) = \{x\}$

- $\text{FV}(t_1 t_2) = \text{FV}(t_1) \cup \text{FV}(t_2)$

- $\text{FV}(\lambda x. t) = \text{FV}(t) - \{x\}$

$$\text{FV}((\lambda x. x_1) x_2) \quad \begin{matrix} t_1 \\ t_2 \\ \text{free} \end{matrix} \quad \begin{matrix} \text{https://powcoder.com} \\ \text{Add WeChat powcoder} \\ \text{Free} \end{matrix} \quad \text{FV}(\lambda y. ((\lambda x. (x, y)) x_2))$$

$$= \text{FV}(\lambda x. x_1) \cup \text{FV}(x_2)$$

$$= \text{FV}((\lambda x. (x, y)) x_2) - \{y\}$$

$$= \text{FV}((x_1) - \{x\}) \cup \{x_2\}$$

$$= (\text{FV}(\lambda x. (x, y)) \cup \text{FV}(x_2)) - \{y\}$$

$$= \emptyset \cup \{x_2\}$$

$$= (\text{FV}(x, y) - \{x\} \cup \{x_2\}) - \{y\}$$

$$= \{x_2\}$$

$$= (\{y\} \cup \{x_2\}) - \{y\}$$

x_1 is bound

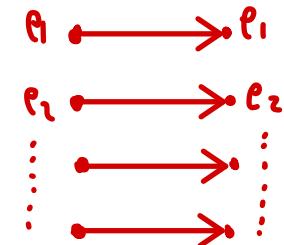
$$= \{x_2\}$$

α -Reduction (Informal)

Identity function: $\lambda x. x$

is the same as $\lambda y. y$ and $\lambda z. z$ etc.

Assignment Project Exam Help



Observation: the name of a parameter is irrelevant in λ -calculus

Add WeChat powcoder

Analogy in C:

Is same as int f (int x) { return x; }

Is same as int f (int y) { return y; }

Is same as int f (int z) { return z; }

α -Reduction

Observation: the name of a parameter is irrelevant in λ -calculus

Assignment Project Exam Help

<https://powcoder.com>

$(\lambda x.(x\ x)) =? (\lambda y.(y\ y))$

$(\lambda x.(x\ x)) =? (\lambda y.\alpha(y))$

$\underline{x} =? y$

$(\lambda \underline{x}.\lambda x.\underline{x}) =? (\lambda \underline{y}.\lambda x.\underline{y})$

α -Reduction (formal)

$\lambda x.t = \lambda y.t\{y/x\}$ when $y \notin FV(t)$

where $t\{y/x\}$ means substitute **all free** x in t with y

$$FV(x_1 \cup x_2)$$

$$= FV(x_1) \cup FV(x_2)$$

$$= \{x_1, x_2\}$$

<https://powcoder.com>
 $(\lambda x.(x, x)) = (\lambda y. y, y)$

Add WeChat powcoder
 $(\lambda x. x, x) \neq (\lambda y. x, y)$

$$x \neq y$$

$$(\lambda x. (\lambda x^t. x)) \neq (\lambda y. \lambda x. y)$$

$$= \lambda y. ((\lambda x^t. x) \{y/x\})$$

$$= \lambda y. (\lambda x. x)$$

when $y \notin FV(\lambda x. x) \checkmark$

$$FV(\lambda x. x)$$

$$= FV(x) - \{x\}$$

$$= \{x\} - \{x\}$$

$$= \emptyset$$

#4

Functional Programming and Scheme

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Free vs. Bound Variable

- $\text{FV}(x) = \{x\}$
- $\text{FV}(t_1 t_2) = \text{FV}(t_1) \cup \text{FV}(t_2)$
- $\text{FV}(\lambda x. t) = \text{FV}(t) - \{x\}$

Example: For the following term, connect all bound variables to their definitions with lines
Add WeChat powcoder

$$\begin{aligned}\text{FV}(t) &= \text{FV}((x z) (\lambda y. (y x))) - \{x\} \\ &= \text{FV}(x, z) \cup \text{FV}(\lambda y. (y x)) - \{x\} \\ &= \{x_1, z\} \cup \{x_2\} - \{x\} \\ &= \{z\}\end{aligned}$$

$$t = (\lambda x. \underbrace{((x z))}_{t_1} (\lambda y. \underbrace{(y x)}_{t_2})))$$

$$\begin{aligned}\text{FV}(x, z) &= \text{FV}(\lambda y. (y x)) \\ &= \text{FV}(y) \cup \text{FV}(x) - \{y\} \\ &= \{y, x\} - \{y\} \\ &= \{x\}\end{aligned}$$

() : optional

Capture-Avoiding Substitution

$t\{y/x\}$: substitute **all free** x in t with y when y is not captured in t (i.e., there is no λy in t)

~~Assignment Project Exam Help~~

captured variable check

<https://powcoder.com>

Examples

$$x\{y/x\} = y \quad \text{Add WeChat (powcoder)}$$

$$(\lambda x. x)\{y/x\} = (\lambda x. x) \quad (\lambda \underline{y}. \underline{x})\{y/x\} \neq (\lambda \underline{y}. \underline{y})$$

$$\text{FV}(\lambda x. x)$$

$$= \text{FV}(x) - \{x\}$$

$$= \{x\} - \{x\}$$

$$= \emptyset$$

captured

≠

$$(\lambda y. x)$$

free

α -Reduction

Replacing all bound variables with their definition gives the same term

[Assignment](#) [Project](#) [Exam](#) [Help](#)
 $(\lambda x. x) = (\lambda y. y)$ $(\lambda x. x \ x) = (\lambda y. y \ y)$
 $\underline{(\lambda x. x \ x)} \neq (\lambda y. x \ y)$ $x \neq y$
https://powcoder.com
Add WeChat powcoder

More formally:

$$\underline{\lambda x. t} = \lambda y. \underline{t\{y/x\}} \quad \underline{\text{when } y \notin \text{FV}(t)}$$

Subtle point: what if y is captured in t ? Use α -reduction to rename the captured y in t

$\hookrightarrow \lambda y \in t \quad \text{rename } y \text{ as } z$

β -Reduction (Informal)

Identity function: $\lambda x. \underline{x}$

$(\lambda x. x) y = y$

Assignment Project Exam Help
Input Output
Observation: we can substitute the formal parameter with the true parameter
Add WeChat powcoder

Analogy in C:

Abstraction: int f (int x) { return x }

Application: f(y) evaluates to y

β -Reduction

The key reduction rule in λ calculus

$(\lambda x. t_1) t_2 \Rightarrow t_1[x \rightarrow t_2]$ Assignment Project Exam Help

buddy

<https://powcoder.com>

Capture-avoiding
Add WeChat powcoder
substitution

β -Reduction Example

Example 1: $(\lambda x. \underline{(x \ x)}) \ y$

β -Reduction

$$(\lambda x. t_1) t_2 = t_1\{t_2/x\}$$

= $(\lambda x. \{y/y\}) \ y$ (β -reduction)
= $(y \ y)$ since y is not captured in $(x \ x)$
<https://powcoder.com>

Add WeChat powcoder

$FV(x_1 x_2)$

= $FV(x_1) \cup FV(x_2)$

= $\{x_1\} \cup \{x_2\}$

= $\{x_1, x_2\}$

β -Reduction Example

Example 2: $(\lambda x.(\lambda y.(x y))) \underline{\underline{y}}$

β -Reduction

$$(\lambda x. t_1) t_2 = t_1\{t_2/x\}$$

$\vdash_{\beta} (\lambda y. (x y)) \{y/x\}$ ← captured variable check here
check y is not captured in the body $\lambda y. (x y)$

$\vdash_{\alpha} (\lambda z. (x z)) \{y/x\}$ failed b/c there is a y

$$(x y) \{z/y\}$$

free

$$:(x z)$$

$$:\lambda z. (y z)$$

$$\text{FV}(\lambda z. (x z))$$

= ...

$$= \{x\}$$

Useful Rule

We say a term t is ***closed*** if it contains no free variable

Assignment Project Exam Help
Examples: $(\lambda x. x)$ $(\lambda x. \lambda y. x y)$

<https://powcoder.com>

Rule: for a capture ~~Add We Check pos~~ substitution

$$t_1\{t_2/x\}$$

the subtle captured-variable check is vacuously true when t_2 is closed \Rightarrow skip captured variable check

Another Useful Rule

$$(((\lambda x_1. \lambda x_2. \dots \lambda x_n. t) t_1) t_2) \dots t_n = (\lambda x_1. x_2. \dots x_n. t) t_1 t_2 \dots t_n = t\{t_1/x_1\} \dots \{t_n/x_n\}$$

when $t_1 t_2 \dots t_n$ are all closed terms

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

Add WeChat powcoder

Evaluation

An ***evaluation*** of a λ term is a sequence

$t_1 = t_2 = t_3 = \dots$

Assignment Project Exam Help

where each step is either an α -reduction
or a β -reduction

Add WeChat powcoder

<https://powcoder.com>

Evaluation Order

No reduction order is specified in classical λ -Calculus

If evaluation terminates, any order gives same result

$$\begin{aligned} & (\lambda x. t) \quad \text{https://powcoder.com} \quad \lambda x. t \quad t \\ & (\lambda x. (\lambda y. x) z) u \quad (\lambda x. (\lambda y. x) z) u \\ = & (\lambda y. u) z \quad \text{Add WeChat powcoder} \quad = (\lambda x. x) u \\ = & u \quad = u \end{aligned}$$

#5

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Assignment Project Exam Help

Is the λ -Calculus Turing complete?

<https://powcoder.com>

Add WeChat powcoder

Encoding: Boolean

Booleans

Shorthand for
 $\lambda x.(\lambda y.x)$

TRUE $\triangleq \lambda x.y.x$ FALSE $\triangleq \lambda x.y.y$

Encoding of “if”? <https://powcoder.com>

Goal: IF $b\ t\ f \triangleq \begin{cases} t & \text{when } b \text{ is TRUE} \\ f & \text{when } b \text{ is FALSE} \end{cases}$

Definition IF $\triangleq \lambda b\ t\ f.\ (b\ t\ f)$

Encoding: Boolean

Booleans

$$\text{TRUE} \triangleq \lambda x. y. x \quad \text{FALSE} \triangleq \lambda x. y. y$$

Assignment Project Exam Help
<https://powcoder.com>
Encoding of “and”?

Goal: AND $b_1 b_2 \triangleq \begin{cases} \text{TRUE when } b_1, b_2 \text{ are both TRUE} \\ \text{FALSE otherwise} \end{cases}$

Definition AND $\triangleq \lambda b_1 b_2. (b_1 (b_2 \text{ TRUE FALSE}) \text{ FALSE})$

Check that AND TRUE FALSE = FALSE (Note 2)

\triangleq = defined as

Is lambda calculus Turing complete?

Boolean Encoding

$$\text{TRUE} \triangleq \lambda x. y. x$$

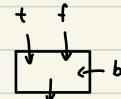
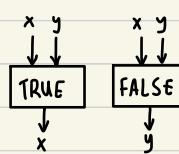
$$\text{FALSE} \triangleq \lambda x. y. y$$

$$\text{AND} \triangleq \lambda b_1. b_2. (b_1(b_2 \text{ TRUE FALSE})) \text{ FALSE}$$

$$\text{IF} \triangleq \lambda b. t. f. \text{butuf}$$

$$\text{OR} \triangleq \lambda b_1. b_2. (b_1 \text{ TRUE}(b_2 \text{ TRUE FALSE}))$$

$$\triangleq \lambda b_1. b_2. (b_1 \text{ TRUE } b_2)$$



\wedge	T	F
T	T	F
F	F	F

\vee	T	F
T	T	T
F	T	F

Ex: IF TRUE TRUE FALSE

$$= (\lambda b. t. f. b. t. f) \text{ TRUE } \text{ TRUE } \text{ FALSE} \quad (\text{by def of IF})$$

$$= \beta(\lambda b. t. f) \text{ TRUE } (\lambda b. t. f) \text{ TRUE } \text{ FALSE}$$

$$= \text{TRUE } \text{ TRUE } \text{ FALSE}$$

$$\cdot (\lambda x. y. x) \text{ TRUE } \text{ FALSE} \quad (\text{by def of TRUE})$$

$$= \beta(\lambda x. y. x) \text{ TRUE } (\lambda x. y. x) \text{ FALSE}$$

$$= \text{TRUE}$$

$x \wedge y$ infix
AND $x \ y$ prefix

↓ is a function

① Ex: If b_1 ($\text{IF } b_2 \text{ TRUE FALSE}) \text{ FALSE}$

$$= \dots$$

$$\textcircled{2} \Rightarrow b_1 (b_2 \text{ TRUE FALSE}) \text{ FALSE}$$

$$\textcircled{3} \Rightarrow b_1. b_2 \text{ FALSE}$$

$$b_2 \text{ TRUE FALSE}$$

is the same as b_2

b^2 is either TRUE or FALSE

If b_2 is TRUE, $b_2 \text{ TRUE FALSE} : (\lambda x. y. x) \text{ TRUE } \text{ TRUE}$

$$= \beta(\lambda x. y. x) \text{ TRUE } (\lambda x. y. x) \text{ FALSE}$$

$$= \text{TRUE}$$

If b_2 is FALSE, ...

$$= \text{FALSE}$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

$$(T \wedge F = F)$$

Ex: AND TRUE FALSE

$$\begin{aligned}
 &= (\lambda b_1 b_2. (b_1 (b_2 \text{ TRUE FALSE}) \text{ FALSE}) \text{ TRUE FALSE}) \text{ closed } (\text{def of AND}) \\
 &= \beta (b_1 (b_2 \text{ TRUE FALSE}) \text{ FALSE}) \{ \text{TRUE}/b_1 \} \{ \text{FALSE}/b_2 \} \\
 &= \text{TRUE} \text{ (FALSE TRUE FALSE) FALSE} \\
 &= (\lambda x y. x) (\text{FALSE} \text{ TRUE FALSE}) \text{ FALSE} \quad (\text{def of TRUE}) \\
 &= \beta (x) \{ (\text{FALSE} \text{ TRUE FALSE}) / x \} \{ \text{FALSE} / y \} \\
 &= \text{FALSE} \text{ TRUE FALSE} \\
 &= (\lambda x y. y) \text{ TRUE FALSE} \quad (\text{def of FALSE}) \\
 &= \beta (y) \{ \text{TRUE} / x \} \{ \text{FALSE} / y \} \\
 &= \text{FALSE}
 \end{aligned}$$

Assignment Project Exam Help

Natural Numbers

function composition

$$0 \ \lambda f z. z$$

$$f \circ g \triangleq \lambda x. f(g(x))$$

$$1 \ \lambda f z. f z$$

fold composition

$$2 \ \lambda f z. f(f z)$$

$$f^1 \ 1\text{-fold} \dots f(x)$$

$$n \ \lambda f z. f(f \dots (f z))$$

$$f^2 \ 2\text{-fold} \dots f(f(x))$$

$$\downarrow$$

$$f^n \ n\text{-fold} \underbrace{f(f \dots (f x))}_{n \text{ times}}$$

church #'s

Add WeChat powcoder

$$\text{succ } 0 = 1$$

$$\text{succ } 1 = 2$$

$$\vdots$$

$\text{succ } n = \underline{n+1}$ n has to power of repeating f on z for n times

$$\text{succ } \triangleq \lambda n. \lambda f z. f(n, f, z)$$

input: 1

↑
church #

$$= (\lambda n. \lambda f z. f(n, f, z)) \underline{1} \quad (\text{def. succ})$$

output: n+1

$$= \beta (\lambda f z. f(n, f, z)) \{ \underline{1} / n \}$$

$$= \lambda f z. f(\underline{1}, f, z)$$

$$= \lambda f z. f((\lambda f z. f(z)) f, z)$$

$$= \lambda f z. f(f(z)) \xrightarrow{\text{free}} \text{natural } \#$$

$$= \beta \lambda f z. f((f z) \{ f / f \} \{ z / z \})$$

$$= \lambda f z. f(f z)$$

$$= \underline{2}$$

PLUS $\triangleq \lambda n_1. \lambda n_2. (n_1 \text{ succ } n_2)$

$\lambda n_1. \lambda n_2. (n_2 \text{ succ } n_1)$

$$\begin{aligned} & \overbrace{\quad \quad \quad}^{\wedge 2} \\ & \quad \quad \quad \overbrace{n_1 + 1 + 1 + \dots + 1}^{\wedge n_2} \\ & // \quad \quad \quad \overbrace{n_1 + n_2 + 1 + 1 + \dots + 1}^{\wedge n_1} + n_2 \\ & \quad \quad \quad : \underbrace{(1+) (1+) \dots (1+)}_{n_1} + n_2 \end{aligned}$$

repeat $(1+)$ for $\wedge n_1$
times on $\wedge n_2$

Plus 1 2

\vdots

$\vdash 3$ (see in note 2)

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

#6

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Church Encoding

Natural numbers

*n-fold
composition*

Assignment Project Exam Help

Church numerals: $\underline{n} \triangleq \lambda f z. f^n z$

Note: \underline{n} is the encoding of number n

$$\underline{0} \triangleq \lambda f z. f^0 z = \lambda f z. z$$

$$\underline{1} \triangleq \lambda f z. f^1 z = \lambda f z. (f z)$$

$$\underline{2} \triangleq \lambda f z. f^2 z = \lambda f z. (f (f z))$$

...

Church Encoding

Natural numbers

$$\underline{n} \triangleq \lambda f z. f^n z$$

Assignment Project Exam Help
Encoding of “+ 1”?

Goal: SUCC $\underline{n} = \lambda f z. f^{n+1} z$

Definition SUCC $\triangleq \lambda n f z. (f (n f z))$

Church Encoding

Natural numbers

$$\underline{n} \triangleq \lambda f z. f^n z$$

Assignment Project Exam Help
Encoding of “+”? <https://powcoder.com>

Goal: PLUS $\underline{n_1}$ n_2 = $\lambda f z. f^{n_1 + n_2} z$

Definition PLUS $\triangleq \lambda n_1 n_2. (n_1 \text{ SUCC } n_2)$

Church Encoding: Example

Natural numbers

$$\underline{n} \triangleq \lambda f z. f^n z$$

Definition PLUS $\triangleq \lambda n_1 n_2. (n_1 \text{ SUCC } n_2)$

Add WeChat powcoder

Check that PLUS $\underline{1} \underline{2} = \underline{3}$ (Note 2)

#7

Functional Programming and Scheme

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Church Encoding

λ allows us to
repeat f for n times on z

Natural numbers

Definition $\underline{n} \triangleq \lambda f z. f^n z$

Assignment Project Exam Help

Church #

$n \rightarrow$ Natural #

<https://powcoder.com>

Add WeChat  powcoder

$$\triangleq \lambda n_1 n_2. (n_1 f (n_2 f z))$$

$\uparrow \quad \uparrow \quad \uparrow$
 $n_2 f's$ in body

$$(f \underbrace{f(f(f(f(f(\dots (f}_{n_1} z))) \underbrace{f}_{n_2}))$$

$$n_1 + n_2 = \underbrace{| + | + | + \dots + |}_{n_1} n_2$$

Church Encoding: Example

Natural numbers

$$n \triangleq \lambda f z. f^n z$$

$$\frac{3 \times 4 = 12}{\text{MUL } 3 \cdot 4 = 12}$$

Assignment Project Exam Help
repeat $(+n_2)$ for n_1 times on

Definition $\text{MULT} \triangleq \lambda n_1 \lambda n_2. (n_1 \text{ (plus } n_2 \text{)})^n_1$
Add WeChat powcoder

$$3 \times 4 = \underbrace{4 + 4 + 4}_{3 \text{ times}}$$

$$n_1 \times n_2 = \underbrace{n_2 + n_2 + n_2 + \dots + 0}_{n_1 \text{ times}}$$

Multi-Argument Functions

$\lambda x.t$

$SUCC \triangleq \lambda n f z. (f (n f z))$ shorthand for $\lambda n. \lambda f. \lambda z. (f (n f z))$

$PLUS \triangleq \lambda n_1 n_2. (n_1 SUCC n_2)$

Currying: every function is treated as one that takes at most one parameter at one time

Add WeChat powcoder

$(\lambda x_1 x_2 \dots x_n. e)$ is the same as $\lambda x_1. (\lambda x_2. (\dots (\lambda x_n. e)))$

so

$(\lambda x y. t) t_1 t_2$ is the same as
 $((\lambda x. (\lambda y. t)) t_1) t_2$

Named Functions

```
int addOne (int n) {  
    return n+1;  
}  
addOne (addOne(1))
```

Use definition $SUCC \triangleq \lambda n f z. (f (n f z))$

in term $(SUCC (SUCC 1))$?

Assignment Project Exam Help

Syntax: **let name = def in body** (or, **let (name def) body**)

E.g., **let SUCC = $(\lambda n f z. (f (n f z)))$ in $(SUCC (SUCC 1))$**

let name = def in body

is just a shorthand for $(\lambda name. body) def$

$(\lambda svcc. (svcc (svcc 1))) \in (\lambda nfz...)$

Pure vs. Applied λ -Calculus

Pure λ -Calculus: the calculus discussed so far

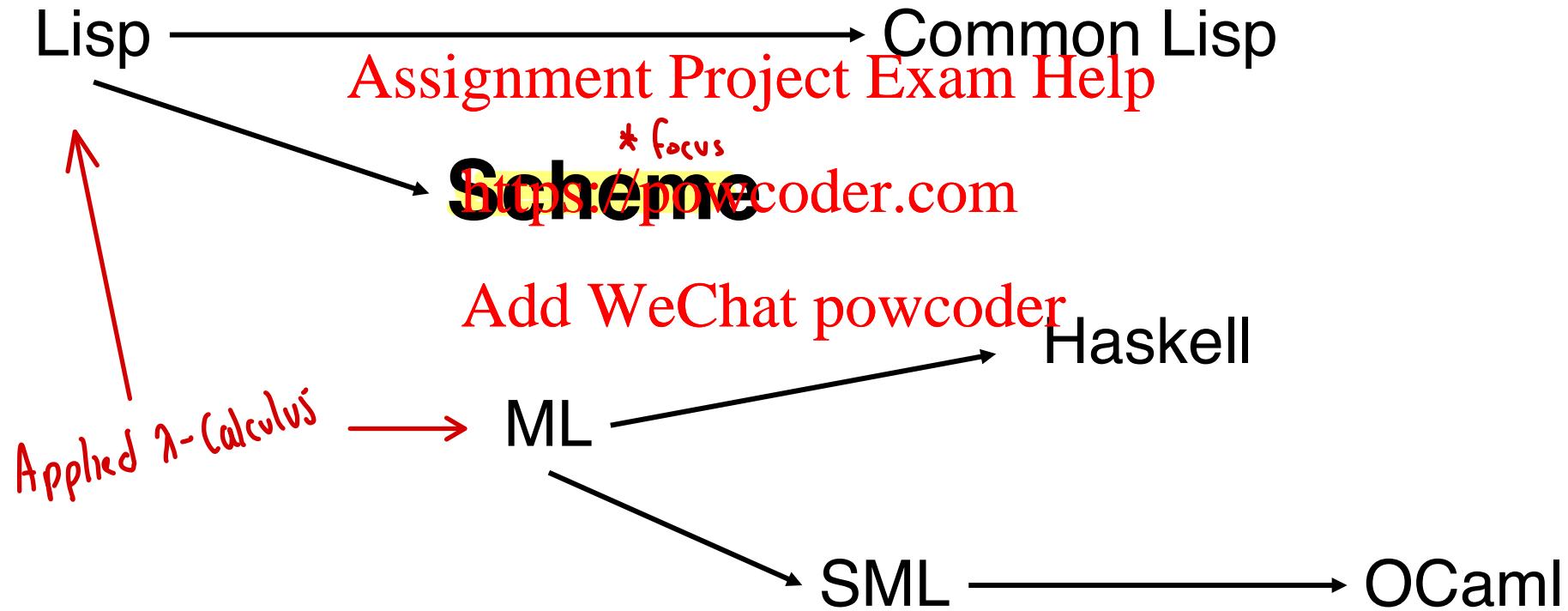
$t ::= \lambda x t | t_1 t_2 | x$
Assignment Project Exam Help

Applied λ -Calculus:

- Built-in values and data structures
(e.g., 1, 2, 3, true, false, (1 2 3))
- Built-in functions
(e.g., +, *, /, and, or)
- Named functions
- Recursion

All features can be encoded
in the pure λ -Calculus!

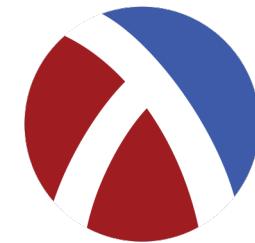
Functional Languages



Running Scheme

Racket: a Scheme-based language

<https://racket-lang.org> Assignment Project Exam Help



DrRacket: IDE for Racket

The screenshot shows the DrRacket IDE interface. At the top, there's a menu bar with tabs for Assignment, Project, Exam, and Help. Below the menu is a toolbar with icons for Untitled, (define ...), Run, Stop, and others. The main window has two panes. The left pane contains the Racket code:#lang racket

(display "hello world!")The right pane shows the output of the code execution:Welcome to DrRacket, version 7.8 [3m].
Language: racket, with debugging; memory limit: 128 MB.
hello world!
> |At the bottom, there are status indicators for memory usage (991.09 MB) and file operations.

Scheme

We focus on the non-imperative subset of Scheme (no assignments, whiles)

Syntax: Assignment Project Exam Help

- Atoms: e.g., 3, #t, #f, “abc”
- Lists: (3 #t “abc”)
- Functions: (sqrt 2), (+ 1 2 4)
- Comments: (+ 1 2) ; evaluate to 3

Read-Eval-Print Interpreter

- (sqrt 2) is the same as (eval '(sqrt 2))

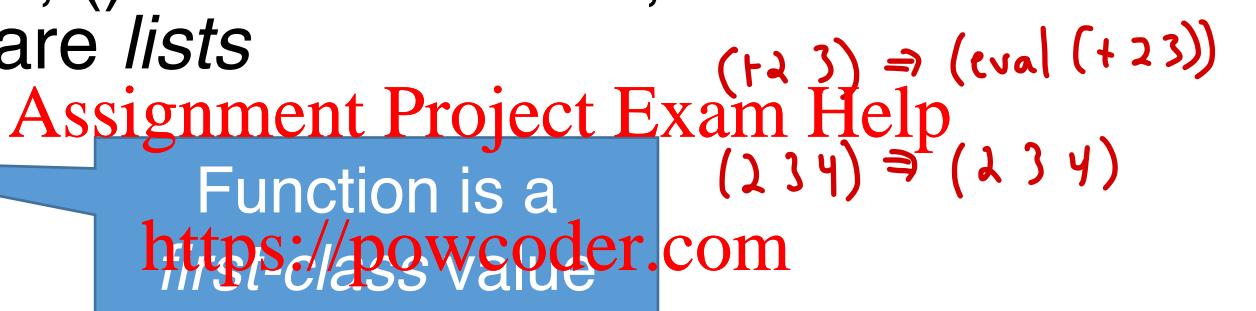
list w/ eval before it

fun
 ↑
 pm“
 ↑
 list

Parenthesis in Scheme

In Scheme, () is used for lists, and both code and data are *lists*

- (+ 2 3)
- '(2 3 4)



Lists are evaluated as function calls

- (+ 2 3)

Evaluates to 5

- ((+ 2 3))

Exception: 5 is not a procedure

- '(2 3 4)

Exception: 2 is not a procedure

(eval(2 3 4)) \Rightarrow error

Lists w/o evaluation: use apostrophe (')

- '(2 3 4)

Returns (2 3 4)

#8

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Parenthesis in Scheme

In Scheme, () is used for lists, and both code and data are *lists*

- (+ 2 3) Assignment Project Exam Help
- (2 3 4) https://powcoder.com

Lists are evaluated as function calls
Add WeChat powcoder

- (+ 2 3) Evaluates to 5
- ((+ 2 3)) Exception: 5 is not a procedure
- (2 3 4) Exception: 2 is not a procedure

Lists w/o evaluation: use apostrophe (')

- '(2 3 4) Returns (2 3 4)

Expressions

Arithmetic:

```
( * 1 2 3 4)  
(+ (* 1 2) (* 2 4))
```

Assignment Project Exam Help

Relational:

```
(< (* 1 2) (+ 1 1))  
(and (> 2 4) #f)
```

Conditional:

Add WeChat powcoder

```
(if (< 2 3) 1 (+ 2 3))  
(* 2 (if (< 3 5) 3 "abc"))
```

Type test:

```
(number? 2) (number? #f)  
(string? "a") (string? 1)  
(boolean? #t) (boolean? 1)
```

Environment

Environment binds identifiers to values

- `+, *, -, /` are identifier bound to values
- Functions are values in Scheme
<https://powcoder.com>

Built-in identifiers in initial environment

Add WeChat powcoder

- `+, sqrt, positive?, max, ...`

Add bindings to the environment

- Key word: `define`

Definition

Built-in identifiers

(Assignment Project Exam Help)

<https://powcoder.com>
“define” introduce global identifiers

Add WeChat powcoder

```
(define pi 3.14159)
(define radius 2)
(* pi (* radius radius))
(define circumference (* 2 pi radius))
circumference
```

Local Definition

Let Expr.:

```
(let ((x 3)
      (y (sqrt 7)))
  (+ x y))
```

Assignment Project Exam Help

General form:

```
(let ((x1 exp1)
      (x2 exp2)
      ...
      (xn expn))
  body_exp)
```

Add WeChat powcoder

x₁, x₂, ..., x_n can be used in body_exp

Local Definition

Let* Expr.:

```
(let* ((x 3)
       (y x))
  (+ x y))
```

Assignment Project Exam Help

General form:

<https://powcoder.com>

Add WeChat powcoder
...
(x11 exp1)
body_exp)

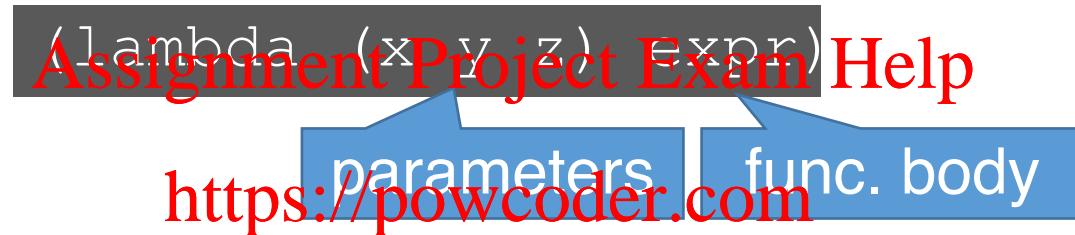
x1 can be used in exp2, exp3, ..., body_exp

x2 can be used in exp3, exp4, ..., body_exp

...

Anonymous Function

Anonymous functions are introduced by lambda



Example

Add WeChat powcoder

```
( (lambda (x) x) 1 )
```

```
( (lambda (x y z) (+ x y z)) 1 2 3 )
```

```
( (lambda (f) (f 2)) (lambda (x) (* 2 x)) )
```

Named Function

```
(define pi 3.14159)
```

Assignment Project Exam Help

In Scheme, function is a first-class value, so
similarly, we can define named functions

Add WeChat powcoder

```
(define f (lambda (x) (+ x 1)))  
(f 1)
```

```
(let ((f (lambda (x) (+ x 1)))  
      (g (lambda (y) (* y 2))))  
  (+ (f 1) (g 2)))
```

Named Function

In Scheme, named functions can be introduced in a more concise way:

Assignment Project Exam Help

```
(define (f x) (+ x 1))
```

<https://powcoder.com>

func. name parameter

Add WeChat powcoder

is the same as

```
(define f (lambda (x) (+ x 1)))
```

Named Function

Name

Parameter

```
(define (sqr x) (* x x))  
Assignment Project Exam Help  
(sqr 5)  
(define (area x y) (* x y))  
https://powcoder.com  
(area 5 10)
```

Add WeChat powcoder

Common use:

```
(define (f x y z)  
  (let ((x1 exp)  
        (x2 exp))  
    f' s body))
```

Function Examples

```
(define (test x)
  (cond ((number? x) "num")
        ((string? x) "str")
        (Assignment Project Exam Help)
        (else "other")))
```

<https://powcoder.com>

```
(define (abs x)
  (cond ((< x 0) (- x))
        (else x)))
```

Add WeChat powcoder

```
(define (factorial n)
  (if (= n 0) 1
      (* n (factorial (- n 1))))))
```

Scheme: Key Points

Evaluation of expressions (e0 e1 e2 e3)

Assignment Project Exam Help
Key words define, if, cond, '
<https://powcoder.com>

No static type system Add WeChat powcoder

```
(define (f x) (+ x "abc"))
```

```
(* 2 (if (< 3 5) 3 "abc"))
```

Scheme: Key Points

No assignments, no iterations (loops)

Assignment Project Exam Help

All variables are immutable (mathematical symbols)

<https://powcoder.com>

Add WeChat powcoder

#9

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Recursion and Induction

Recursion

- Computationally: a procedure/function calls itself
Assignment Project Exam Help
- Semantically: defining a computation/value inductively
<https://powcoder.com>

Induction

Add WeChat powcoder

- Natural number: i) $0 \in \mathbb{N}$ ii) if $n \in \mathbb{N}$, then $(n + 1) \in \mathbb{N}$
- Inductive proof that property P holds on any $n \in \mathbb{N}$:
 - i) for 0, P holds
 - ii) if $n \in \mathbb{N}$ and P holds on n, then P holds on $(n + 1) \in \mathbb{N}$

Factorial

$$n! = n * (n - 1) * \cdots * 1$$

Assignment Project Exam Help
Inductive definition

- Factorial: <https://powcoder.com>

- i) $0! = 1$ Add WeChat powcoder
- ii) $n! = n * (n - 1)!$

```
(define (factorial n)
  (if (= n 0) 1
      (* n (factorial (- n 1))))))
```

Fibonacci Number

Numbers in the following sequence:

1 1 2 3 5 8 13 21

Assignment Project Exam Help

Inductive definition

<https://powcoder.com>

- $\text{fib}(1) = 1$
- $\text{fib}(2) = 1$
- $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$

```
(define (fib n)
  (cond ((= n 1) 1)
        ((= n 2) 1)
        (else (+ (fib (- n 1)) (fib (- n 2)))))))
```

List

A list is a sequence of expressions in parentheses

(1 2 3)

(sqrt x)

(x 1 "abc")

Assignment Project Exam Help

Programs are just lists interpreted as code

<https://powcoder.com>

A list is either Add WeChat powcoder

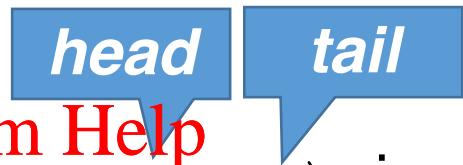
- Empty: ()
- or non-empty: it has a **head** and a **tail**, where
 - the **head** can have any type
 - the **tail** is itself a list

Inductive definition!

List Construction

Inductive definition

- () is a list
- If e is an expression, xs is a list, $(\text{cons } e \ xs)$ is a list



Assignment Project Exam Help

<https://powcoder.com>

“`cons`” is a built-in function for constructing lists

Add WeChat powcoder

(1 2 3 4)

is equivalent to

(`cons` 1 (`cons` 2 (`cons` 3 (`cons` 4 '()))))

List Destruction

↑
contents of address register
→ contents of data register

car and cdr are the destructors for lists:

if xs is a non-empty list, then

Assignment Project Exam Help

(car xs) is the head

(cdr xs) is the tail

Add WeChat powcoder

Algebraically:

$$(\text{car } (\text{cons } x \ xs)) = x$$

$$(\text{cdr } (\text{cons } x \ xs)) = xs$$

List Destruction

(car ' (1 2 3)) = 1, which is the same as

(car (cons 1 ' (2 3)))) = 1
Assignment Project Exam Help

(cdr ' (1 2 3)) = (2 3), which is the same as

(cdr (cons 1 ' (2 3)))) = (2 3)

List

(1 2 3)

(sqrt x)

(x 1 "abc")

Assignment Project Exam Help

“cons” adds an element to a list

<https://powcoder.com>

“car” returns the head of a list

Add WeChat powcoder

“cdr” returns the tail of a list

“append” concatenates two lists

append ' (1 2 3) ' (4 5 6)

Additional List Functions

list creates a list from its arguments

(list 1 2 3)

Assignment Project Exam Help

null? checks if a list is empty

Add WeChat powcoder

list? checks whether something is a list

cadr the second item in a list

caar, cdar, cddr ...

List Example

Define `lstSum`, which returns the sum of a num list

Assignment Project Exam Help

Inductive definition

i) `'(): 0` <https://powcoder.com>

ii) `(cons head tail): head + lstSum(tail)`

Add WeChat powcoder

```
(define (lstSum lst)
  (if (null? lst) 0
      (+ (car lst) (lstSum (cdr lst))))))
```

$(\text{lstSum}'(1 2 3)) = 6$ $(\text{lstSum}'(2 4 6 8)) = 20$

Recursion Over Lists

Assignment Project Exam Help

```
(define (f l) https://powcoder.com
  (if (null? l) (base-case)
      Add WeChat powcoder
      (recursive-case, using (car l), (cdr l))))
```

Equality Test

equal: return #t if two expressions have the same structure and content

Assignment Project Exam Help

```
(equal? 1 1)
```

<https://powcoder.com>

```
(equal? '(1 2) '(1 2))
```

Add WeChat powcoder

```
(equal? 5 '(5))
```

```
(equal? '(1 2 3) '(1 (2 3))))
```

```
(equal? '(1 2) '(2 1))
```

List Example

Define `subs`, which takes `a` `b` `l`, and replaces `a` with `b` in `l`

Assignment Project Exam Help

```
(define (subs a b l)
  base case (if (null? l) '()
    (let ((rest (subs a b (cdr l)))))
      (if (equal? (car l) a)
          (cons b rest)
          (cons (car l) rest)))))
```

(subs 1 2 '(1 1 2 2 3 3))

: '(2 2 2 2 3 3)

#10

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Higher-Order Functions

In Scheme, function is a first-class value

(Functions can go wherever expressions go)

Assignment Project Exam Help

- Higher Order Functions
 - Function can be used as an argument
 - Function can be used as a return value

Function as Argument

Repeat any function twice?

```
(define (double x) (* x 2))  
(define (quad x) (double (double x)))
```

```
(define (square x) (* x x))
```

```
(define (fourth x) (square (square x)))
```

More reusable code

```
(define (twice f x) (f (f x)))
```

```
(twice double 2)
```

```
(twice quad 2)
```

can be simplified

Examples w/o Higher-Order Functions

Find all numbers > 5 in a list

```
newlist = empty
Assignment Project Exam Help
foreach (elem in lst)
    if (elem > 5)
        add elem to newlist
Add WeChat powcoder
```

Find all strings in a list

```
newlist = empty
Assignment Project Exam Help
foreach (elem in lst)
    if (elem is String)
        add elem to newlist
```

Filter

takes each element in list
either return for #t or #f

filter (f) (l): return elements for which f returns #t

Assignment Project Exam Help

f ↓

↓

↓

https://powcoder.com

□ x x ... □

Add WeChat powcoder

return value

Examples w Higher-Order Functions

filter f l: return elements for which f returns #t

Assignment Project Exam Help ' (710)

```
(filter (lambda (x) (> x 5)) ' (1 4 7 10))
```

Add WeChat powcoder #if #f

```
(filter string? ' (1 "1" 2 "2" 3 "3"))
```

C: 1 4 7 10
↓ ↓ ↓ ↓
#f {()#t {()#t {()#t

Examples w/o Higher-Order Functions

Multiple 2 to each element in a list

```
newlist = empty
Assignment Project Exam Help
foreach (elem in lst)
    add elem*2 to newlist
https://powcoder.com
```

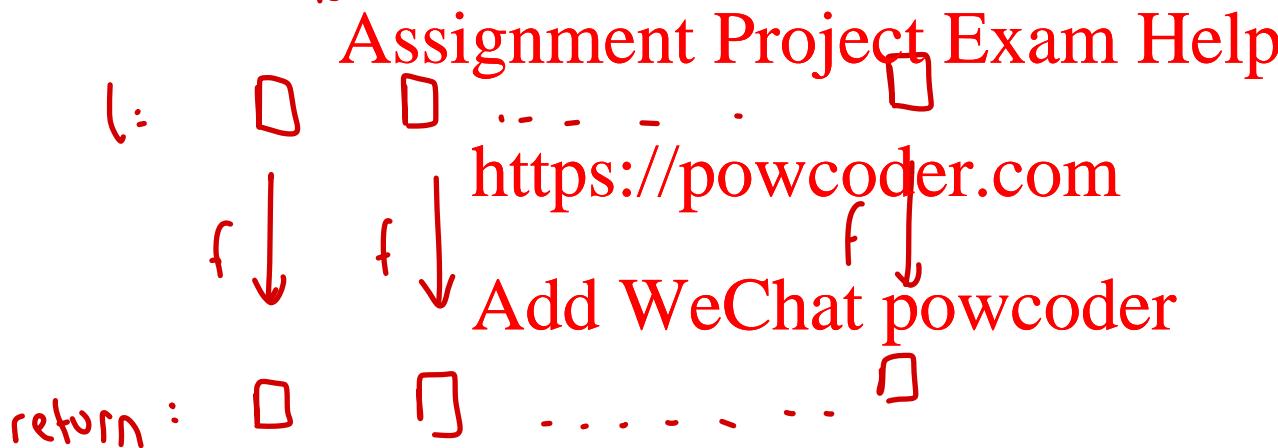
Square each element in a list

```
newlist = empty
Assignment Project Exam Help
foreach (elem in lst)
    add elem*elem to newlist
https://powcoder.com
```

Map

takes element in l
and new element in result

map f l : applies function f to each element of list l



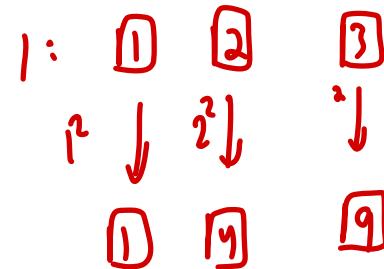
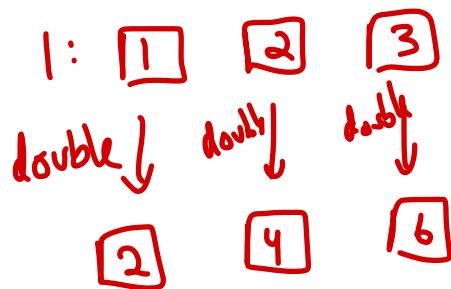
Examples w Higher-Order Functions

map f l : applies function f to each element of list l

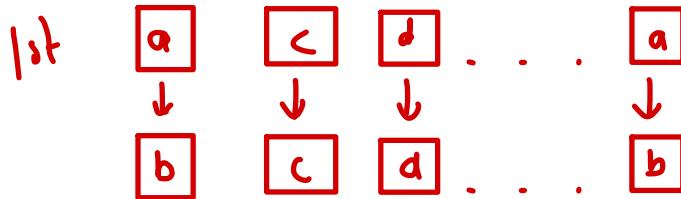
Assignment Project Exam Help

(map double '(1 2 3)) '(2 4 6)
<https://powcoder.com>

(map square '(1 2 3)) '(1 4 9)



Map



$\lambda x. (\text{if } (\text{equal? } x \ a) \ b \ x)$

In previous lecture, we had:

```
(define (subs a b lst)
  (if (null? lst) '()
      (let ((rest (subs a b (cdr lst))))
        (if (equal? (car lst) a)
            (cons b rest)
            (cons (car lst) (subs a b (cdr rest)))))))
```

Using map, we have

```
(define (subs a b lst)
  (map (lambda (c)
          (if (equal? a c) b c))
       lst))
```

Map on Multiple Lists

map f 11 12 13: applies function f to elements at the same position of list 11, 12, 13 (with same length)

Assignment Project Exam Help

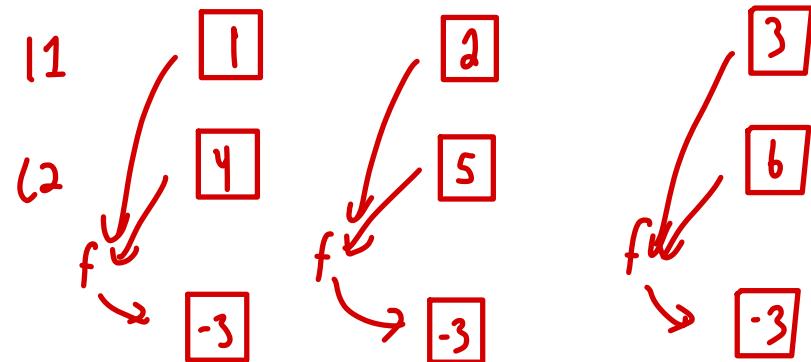
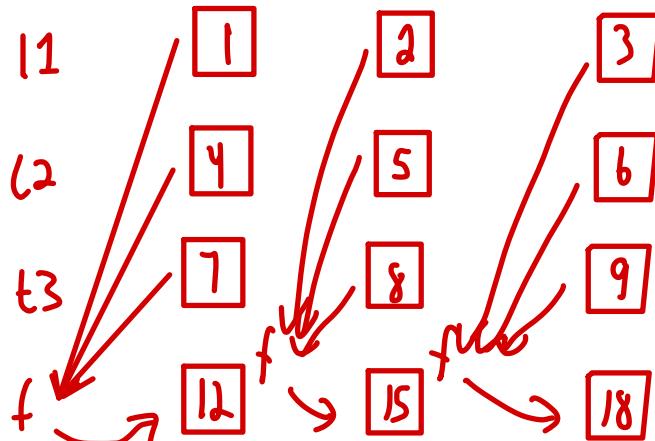
1st

```
(map + ' https://powcoder.com' (1 2 3) (4 5 6) (7 8 9))
```

1st

Add WeChat powcoder

```
(map (lambda (x y) (- x y)) ' (1 2 3) ' (4 5 6))
```



#11

Functional Programming and Scheme

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Higher-Order Functions

In Scheme, function is a first-class value

(Functions can go wherever expressions go)

Assignment Project Exam Help

- Higher Order Functions
 - Function can be used as an argument
 - Function can be used as a return value

Examples w/o Higher-Order Functions

Compute the sum of a list

```
init = 0
Assignment Project Exam Help
foreach (elem in lst)
    init += elem
https://powcoder.com
```

Difference

Add WeChat ^{difference} powcoder

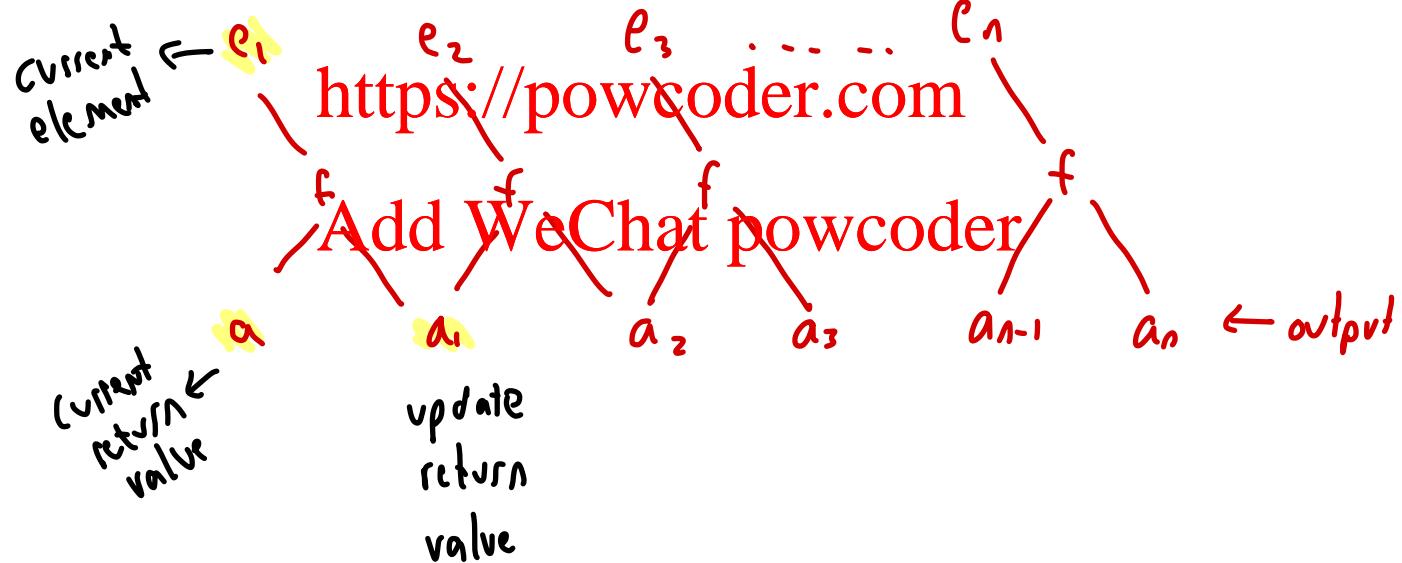
Compute the product of a list

```
init = 1
Assignment Project Exam Help
foreach (elem in lst)
    init *= elem
https://powcoder.com
```

Foldl

foldl f a ($e_1 \ e_2 \ e_n$): returns

$f \ e_n \ (\dots \ (f \ e_2 (f \ e_1 a)) \dots)$



Examples with Higher-Order Functions

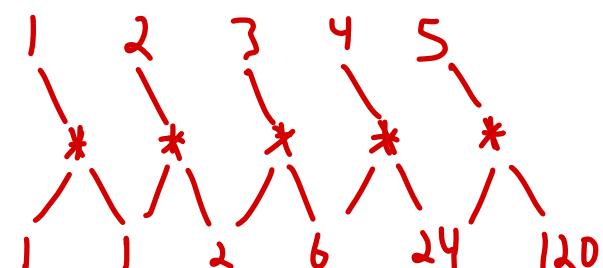
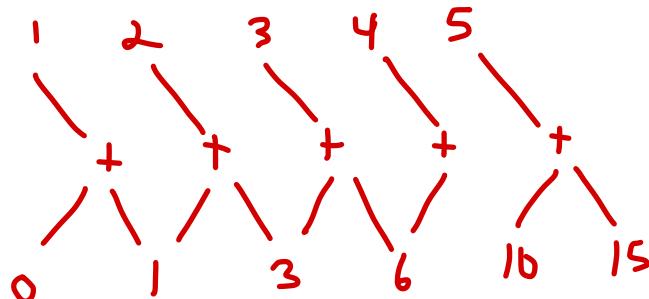
`foldl f a (e1 e2 en)`: returns

`f en (... (f e2 (f e1 a)))`

Assignment Project Exam Help

(`foldl + 0 [1 2 3 4 5]`)

(`foldl * 1 [1 2 3 4 5]`)



Foldl

In previous lecture, we had:

```
(define (lstSum lst)
  (if (null? lst) 0
      (+ (car lst) (lstSum (cdr lst)))))
```

Add WeChat powcoder
Using foldl, we have

```
(define (lstSum lst) (foldl + 0 lst))
```

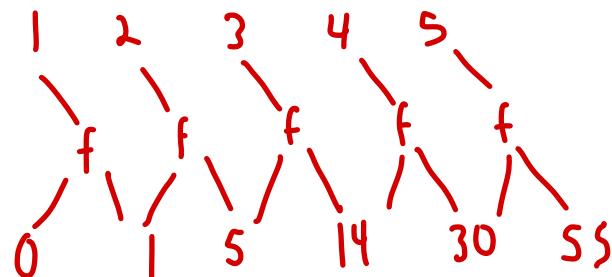
Examples with Higher-Order Functions

`foldl f a (e1 e2 en)`: returns

$f e_n (\dots (f e_2 (f e_1 a) \dots))$

(`foldl (lambda (x a) (+ a (* x x)))`

0 '(1 2 3 4 5))



$$\begin{aligned} & f 1 0 \\ & = 0 + 1^2 = 1 \end{aligned}$$

$$\begin{aligned} & f 2 0 \\ & = 1 + 2^2 = 5 \end{aligned}$$

Function as Return Value

In Scheme, function is a first-class value

(Functions can go wherever expressions go)

Assignment Project Exam Help

- Functions as return values

```
https://powcoder.com  
(define (addN n) (lambda (m) (+ m n)))
```

```
((addN 10) 20)
```

```
(+ 10 20)
```

```
(twice (addN 10) 20)
```

```
(twice (+ 10) 20)
```



Currying and Assignment Project Exam Help

Partial Evaluation

Add WeChat powcoder

Multi-Argument Functions

```
(define (add m n) (+ m n))
```

add: Int, Int → Int

Assignment Project Exam Help

<https://powcoder.com>

Pass multiple parameters as a list?

Add WeChat powcoder

```
(add ' (1 2) )
```



add takes 2
parameters

```
(add 1 2)
```



Multi-Argument Functions

```
(define (add m n) (+ m n))
```

add: Int, Int → Int

Assignment Project Exam Help

<https://powcoder.com>

Add 2 to each element in a list?

Add WeChat powcoder

map f l: applies function f to each element of list l

```
(map (add 2) '(1 2 3))
```



Multi-Argument Functions

```
(define (add m n) (+ m n))
```

add: Int, Int → Int

Assignment Project Exam Help

<https://powcoder.com>

Add 2 to each element in a list?

Add WeChat powcoder

map f l: applies function f to each element of list l

```
(map (add 2) '(1 2 3))
```

add takes two parameters

A multi-argument function can only
be used when all parameters are ready!

Currying: Every function is treated as taking at most one parameter

```
(define add (lambda (m n) (+ m n)))
```

add: Int, Int → Int
<https://powcoder.com>

Curried version [Add WeChat powcoder](#)

```
(define (addN n) (lambda (m) (+ m n)))
```

addN: Int → (Int → Int)

also written as: Int → Int → Int (right associative)

Uncurried version

```
(define (add m n)  
      (+ m n))
```

Curried version

```
(define (addN n)  
      (lambda (m) (+ m n)))
```

add: Int, Int → Int

Assignment Project Exam Help

addN: Int → (Int → Int)

<https://powcoder.com>

(add 2 3)

Add WeChat powcoder

(map (add 2)
 ' (1 2 3))



((addN 2) 3)

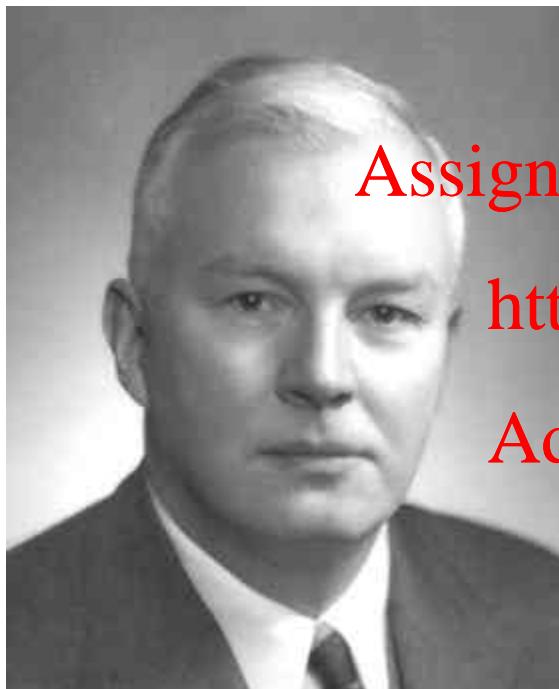


(map (addN 2)
 ' (1 2 3))



Curried form allows partial evaluation

Currying



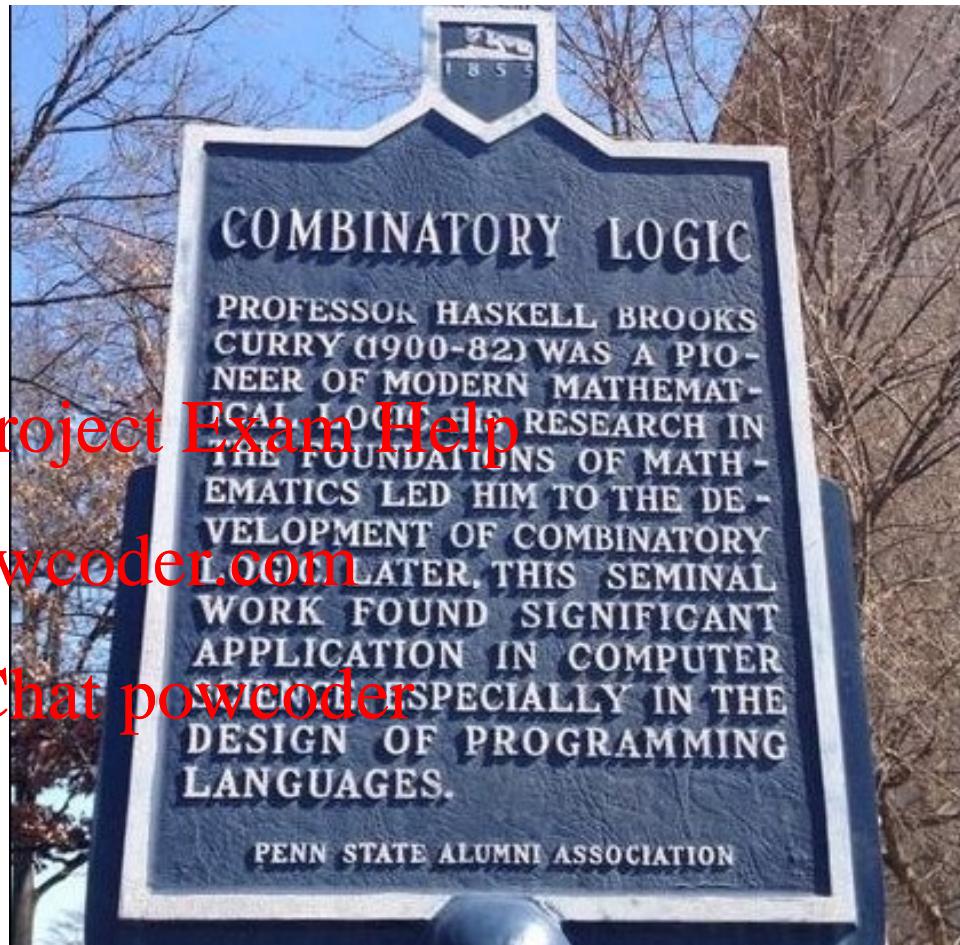
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Haskell B. Curry

Penn State 1929-1966



Outside of McAllister Building

Currying

In terms of lambda calculus,

the curried function of $\lambda x_1 x_2 \dots x_n. e$ is
 $\lambda x_1. (\lambda x_2. (\dots (\lambda x_n. e)))$

<https://powcoder.com>

```
(define (curry2 f)
  (lambda (x)
    (lambda (y)
      (f x y))))
```

Add WeChat powcoder

```
(define (curry3 f)
  (lambda (x)
    (lambda (y)
      (lambda (z)
        (f x y z)))))
```

Partial Evaluation

A function is evaluated with one or more of the leftmost actual parameters

Assignment Project Exam Help

((curry2 add) 2) is a partial evaluation of add
<https://powcoder.com>

Add WeChat powcoder

We can think it as a temporary result,
in the form of a function

Partial Evaluation

A function is evaluated with one or more of the leftmost actual parameters

Assignment Project Exam Help

```
(map ((curry2 add) 2) '(1 2 3))
```



add: $\lambda x y. (+ x y)$ Add WeChat powcoder

(curry2 add): $\lambda x. \lambda y. (+ x y)$

((curry2 add) 2): $\lambda y. (+ 2 y)$

(map ((curry2 add) 2)) '(1 2 3): '(3 4 5)

Uncurrying

In terms of lambda calculus,

the uncurried function of $\lambda x_1.(\lambda x_2.(\dots(\lambda x_n.e)))$ is
 $\lambda x_1 x_2 \dots x_n. e$

[Assignment Project Exam Help](https://powcoder.com)
<https://powcoder.com>

```
(define (uncurry? f)
  (lambda (x y)
    ( (f x) y) )))
```

```
(define (uncurry3 f)
  (lambda (x y z)
    ( ( (f x) y) z) )))
```

Add WeChat powcoder

Uncurrying

In terms of lambda calculus,

the uncurried function of $\lambda x_1.(\lambda x_2.(\dots(\lambda x_n.e)))$ is
 $\lambda x_1 x_2 \dots x_n. e$

[Assignment Project Exam Help](https://powcoder.com)
<https://powcoder.com>

```
(define (uncurry? f)
  (lambda (x y)
    ( (f x) y) )))
```

```
(define (uncurry3 f)
  (lambda (x y z)
    ( ( (f x) y) z) )))
```

Add WeChat powcoder

#12

Assignment Project Exam Help

Syntax <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020



Imperative Language

C, Python

Functional Language

Scheme 1 + 2

OO Language

Java, C++

Assignment Project Exam Help

<https://powcoder.com>

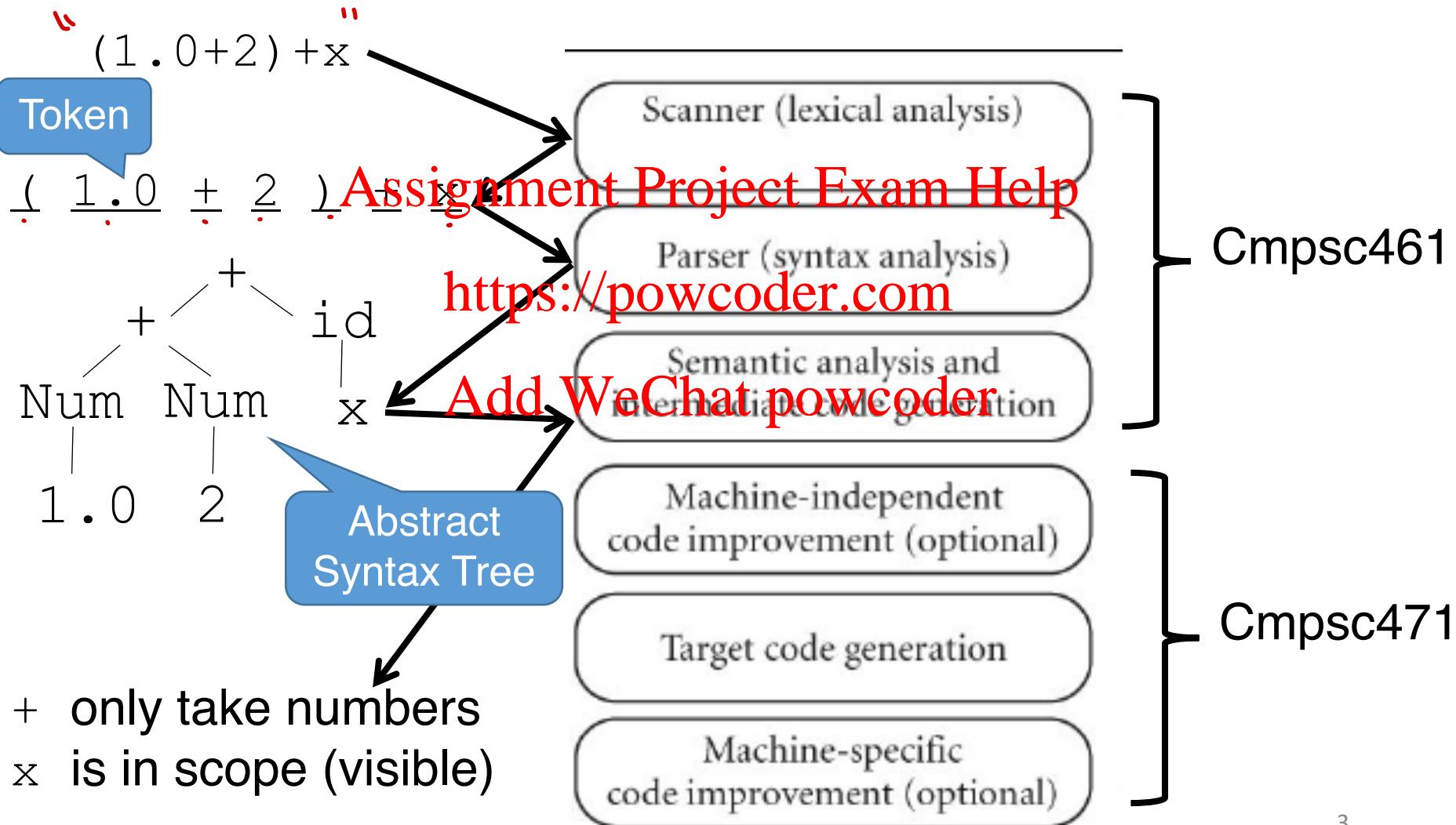
Add WeChat powcoder



3

How does the Magic Box work?

Source Code to Target Code

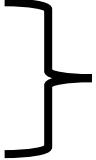


Formal Languages

Language: a set of (legal) strings

Assignment Project Exam Help
Goal: a concise & precise notation for specifying
a language <https://powcoder.com>

Add WeChat powcoder
Four levels of languages [Chomsky]:

- 1. Regular
 - 2. Context-Free
 - 3. Context-Sensitive
 - 4. Unrestricted
- 
- programming languages

Lexical Syntax

tokens

Rules for basic symbols, such as
identifier, literals (e.g., numbers), operators,
keywords
~~Assignment~~ ~~Punctuation~~ Project Exam Help

<https://powcoder.com>

C language: Add WeChat powcoder

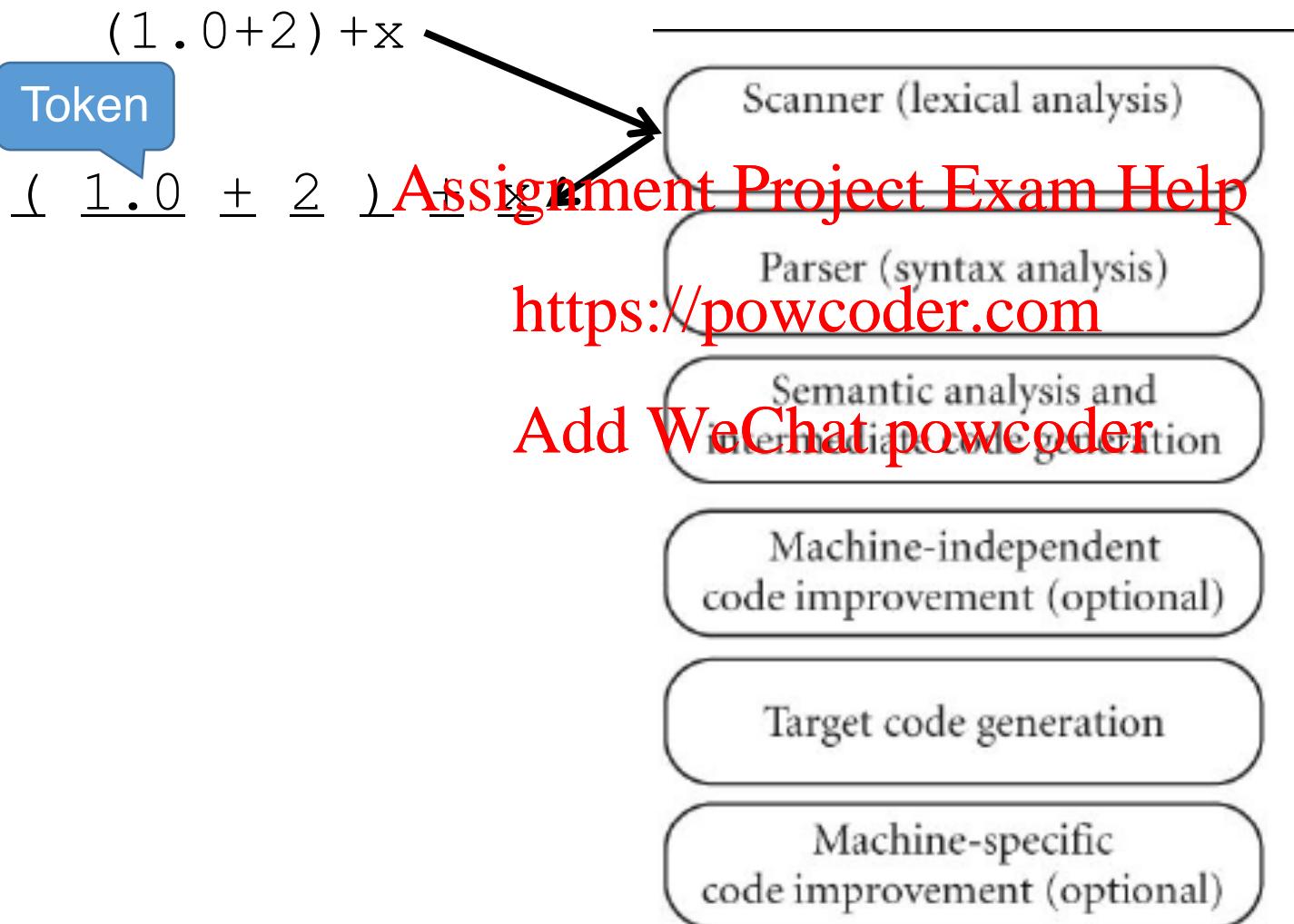
Identifier: letters, digits and underscore '_' only. The first character
must be an underscore or a letter

literals: digits, decimal point, suffix such as "l", "u"

operators: + - * / ...

keywords : if, while, for, int, ...

punctuation: { } [] ; ...



Language of tokens (C)

Identifier: letters, digits and underscore '_' only. The first character must be an underscore or a letter

Assignment Project Exam Help
literals: digits, decimal point, suffix such as "I", "u"

operators: + - * / ... <https://powcoder.com>

keywords : if, while, for, int, ...
Add WeChat powcoder

punctuation: { } [] ; ...

How can we specify these tokens (sets of strings)?

Regular Expression

Definition:

- A character a, b, A, β
- Empty string (ε)
- Concatenation of two RE (e.g., (ab))
- Alternation of two RE, separated by “|” (e.g., (alb))
- Closure (Kleene star) (e.g. (a^*))

$\dots, \alpha\alpha$

Examples

RE	Meaning
a	“a”
ac	“ac”
alc	“a” or “c”
a*	“” or “a” or “aa” or ...
(alb)(cld)	“ac” or “ad” or “bc” or “bd”
(ab)l(cd)	“ab” or “cd”
((alb)c)*	“” or “ac” or “bc” or “acac” or “bcbc” or ...

More Formally...

Regular expression defines *a set of strings* (aka. a language)

Assignment Project Exam Help

$L(R)$: the language defined by RE R

- A character $x: L(x) = \{x\}$
- Empty string $\varepsilon: L(\varepsilon) = \{\varepsilon\}$
- Concatenation: $L(RS) = \{r.s | r \in L(R), s \in L(S)\}$
- Alternation: $L(R|S) = L(R) \cup L(S)$
- Kleene star: $L(R^*) = \{\varepsilon\} \cup L\{R\} \cup L\{RR\} \cup \dots$

String
concatenation

$\text{L}(R)$ means apply L to R

Examples

- A character x : $L(x) = \{"x"\}$
- Empty string ε : $L(\varepsilon) = \{\quad\}$
- Concatenation: $L(RS) = \{r.s | r \in L(R), s \in L(S)\}$
- Alternation: $L(R|S) \vdash L(R) \cup L(S)$
- Kleene star: $L(R^*) = \{""\} \cup L\{R\} \cup L\{RR\} \cup \dots$

<https://powcoder.com>

RE

a $L(a) = \{"a"\}$

ac $L(ac) = \{r.s\}$

alc $L(a|c) = L(a) \cup L(c) = \{"a"\} \cup \{"c"\} = \{"a", "c"\}$

a^* $\{ "", "a", "aa", \dots \}$

$L(\text{RE})$
Add WeChat [powcoder](https://powcoder.com)

$\{"a"\}$

$\{"ac"\}$

$\{"a", "c"\}$

$\{ "", "a", "aa", \dots \}$

Examples

- A character x : $L(x) = \{"x"\}$
- Empty string ε : $L(\varepsilon) = \{\quad\}$
- Concatenation: $L(RS) = \{r.s | r \in L(R), s \in L(S)\}$
- Alternation: $L(R|S) = L(R) \cup L(S)$
- Kleene star: $L(R^*) = \{""\} \cup L\{R\} \cup L\{RR\} \cup \dots$

<https://powcoder.com>

Add WeChat powcoder

RE

$L(\text{RE})$

$$(alb)(cld) = \{r.s | r \in l(alb), s \in l(cld)\} \{“ac”, “ad”, “bc”, “bd”\}$$

$$(ab)l(cd) = \{“\”\} \cup L((alb)_c) \cup L((alb)_c) \{“ab”, “cd”\}$$

$$((alb)c)^* = \{“, “ac”, “bc”, “acac”, “bcbc”, …\}$$

Examples

- A character x : $L(x) = \{"x"\}$
- Empty string ε : $L(\varepsilon) = \{\quad\}$
- Concatenation: $L(RS) = \{r.s | r \in L(R), s \in L(S)\}$
- Alternation: $L(R|S) \vdash L(R) \cup L(S)$
- Kleene star: $L(R^*) = \{""\} \cup L\{R\} \cup L\{RR\} \cup \dots$

<https://powcoder.com>

Add WeChat powcoder

RE

(alb)(cld)

$L(\text{RE})$

{“ac”, “ad”, “bc”, “bd”}

(ab)l(cd)

{“ab”, “cd”}

((alb)c)*

{“, “ac”, “bc”, “acac”,
“bcbc”, ...}

#13

Assignment Project Exam Help

Syntax <https://powcoder.com>

Add WeChat powcoder

CMPSC 461

Programming Language Concepts

Penn State University

Fall 2020

Regular Expression

Definition:

- A character
- Empty string (ε)
- Concatenation of two RE (e.g., (ab))
- Alternation of two RE, separated by “|” (e.g., (alb))
- Closure (Kleene star) (e.g.(a*))

Precedence of RE

The order is (high to low)

Analogy in arithmetic:

- Closure ($*$), then
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Concatenation, then
<https://powcoder.com>
- Alternation

- Exponentiation
- Multiplication
- Addition

Add WeChat powcoder

$$(a^*b) + (c^*a)$$

$$(a) + (b * (c)^e * d)$$

$$(ab)l(cd)$$

$$\rightarrow$$

$$(ab)l(cd)$$

$$(a)l((bc)^*d)$$

$$(a)l(bc)^*d$$

$$\rightarrow$$

$$(a)l(b(c^*)d)$$

$$(a) + ((b*c)^e * d)$$

UNIX Extensions to RE

Extension

Core RE

[abcd]

Assignment Project Exam Help

_ (wild cast)

\rightarrow alblcl... (all char. except new line)

a?

$\varepsilon | a$

a+ { "a", "aa", "aaa", ... }

Add WeChat powcoder
 $a(a^*)$

[a-z]

\rightarrow alblcl... lz

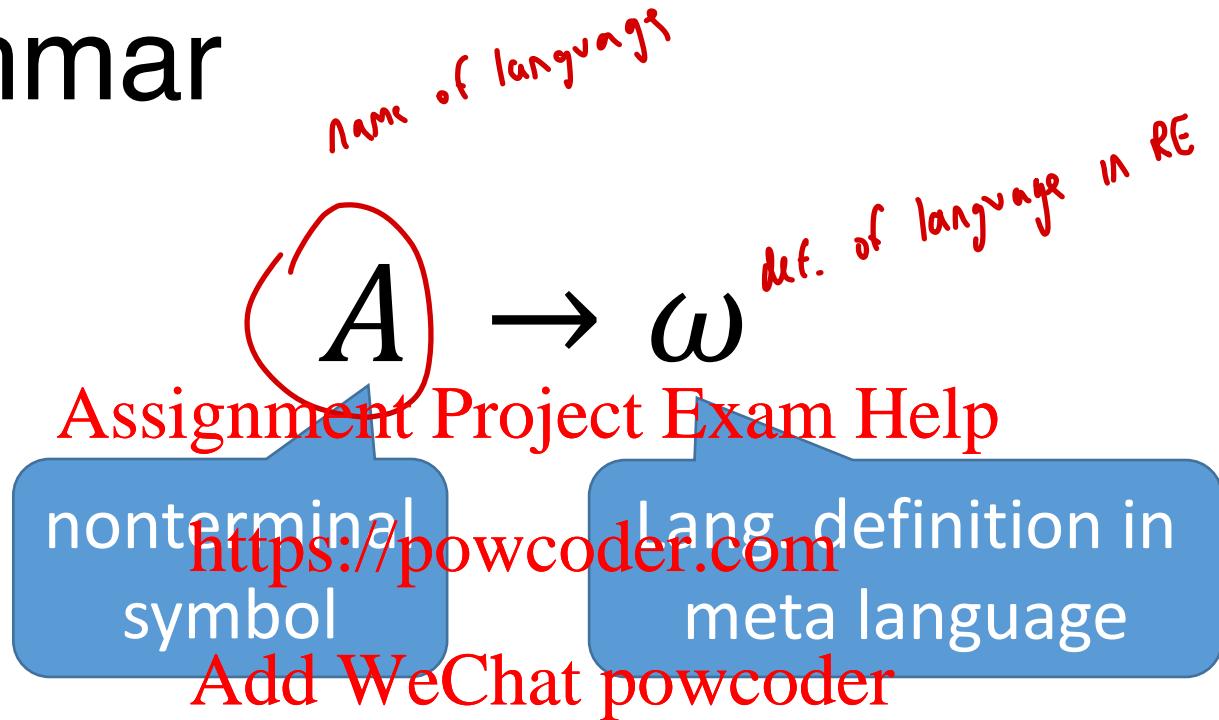
\[\]

\rightarrow [] .

$[10-20]x$

[a-z]
|
of a # of z

Grammar



How to write down a RE grammar?

Language of tokens (C)

Identifier: letters, digits and underscore '_' only. The first character must be an underscore or a letter

numbers: digits, decimal point, suffix such as "l", "u"

operators: + - * / ...

keywords : if, while, for, int, ...

punctuation: { } [] ; ...

<https://powcoder.com>

Grammar

identifier → [_a-zA-Z][_a-zA-Z0-9]*

number → (0| (0|[1-9][0-9]*)(\.[0-9]+)? (l?lu?)

operator → + | - | * | / | ...

keyword → if | while | for | int | ...

punctuation → { | } | \[| \] | ; | ...

Add WeChat powcoder'0'

Is a RE grammar correct?

- Does it cover all legal strings?
Assignment Project Exam Help
- Does it allow any illegal string?
<https://powcoder.com>

Add WeChat powcoder

bit [01]

Examples

RE

bit $\{“0”, “1”\}$ $2(0|1) = \{“0”, “1”\}$ $0|1$

4 bits $[01][01][01][01]$ $(0|1)(0|1)(0|1)(0|1)$

(non-empty) bits $[0]^+$ $(0|1)^+$

(non-empty) Even # of bits $((0|1)(0|1))^+$

frag: Num bet. 0 and 255 <https://powcoder.com> [0-9]

$([01][01])^+$

$[0-9] |$

$[1-9][0-9] |$

$1[0-9][0-9] |$

$2[0-4][0-9] |$

$5[0-5]$

Add WeChat powcoder

$[1-9][0-9]$

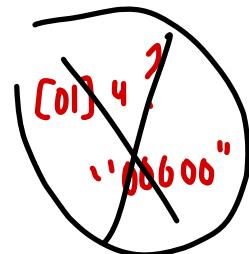
$| 1 [0-9][0-9]$

$| 2 [0-4][0-9]$

$| 25[0-5]$

IP Address $frag.frag.frag.frag.$ Use the RE above

bits with equal 0's and 1's ?



#14

Assignment Project Exam Help
Syntax <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Midterm 1 is rescheduled to Oct. 7th 6:15 to 7:30
Assignment Project Exam Help

HW2 is extended to this Friday midnight
<https://powcoder.com>

Add WeChat powcoder

1st Mideterm covers materials from HW1 to HW3

More details coming later

Scanning with RE

A scanner takes an ordered list of REs

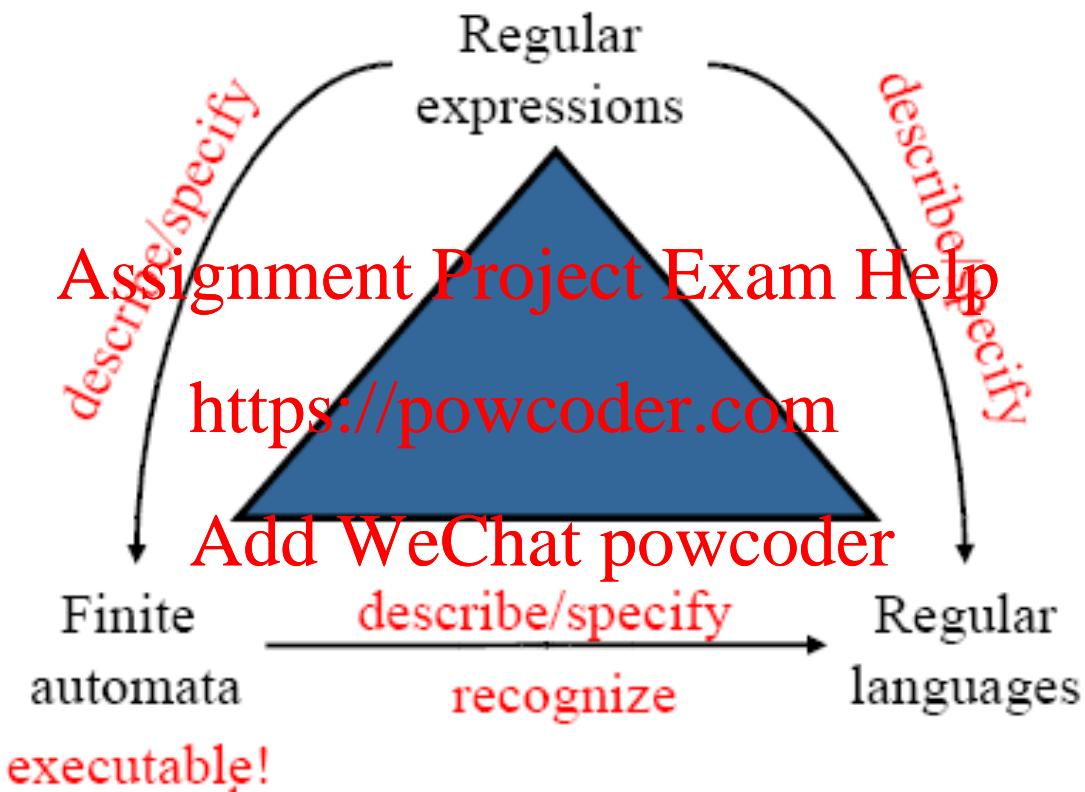
For each RE, read one character at a time and
[Assignment](#) [Project](#) [Exam](#) [Help](#)

- output a token, or
- wait to see next character, or
- use next RE / report an error

```
// hello world
main() /* main */
{for(;;)
    {printf ("Hello World!\n");}
}
```



```
ident("main") lparen rparen lbrace for lparen semi semi
rparen lbrace printf lparen string("Hello World!\n")
rparen semi rbrace rbrace
```



(Not covered
in this class)

Formal Languages

Language: a set of (legal) strings

Assignment Project Exam Help
Goal: a concise & precise notation for specifying
a language <https://powcoder.com>

Add WeChat powcoder
Four levels of languages [Chomsky]:

1. Regular
2. **Context-Free** bits with equal 0's and 1's
3. Context-Sensitive Not a regular language!
4. Unrestricted

Context-Free Grammar (CFG)

A CFG consists of

- A set of *terminals* T (basic alphabet)
Assignment Project Exam Help
<https://powcoder.com>
- A set of *non-terminals* N
- A *start symbol* S (a non-terminal)
- A set of *production rules*
Add WeChat powcoder

$$A \rightarrow \omega$$

nonterminal

string of terminals and
nonterminals, separated by,
' | ' meaning alternation
' ' meaning concatenation

Context-Free Grammar (CFG)

Terminals

- Building block of CFG
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- E.g., letters, digits, or tokens from scanner
<https://powcoder.com>

Non-terminals

- Symbols defined by production rules
- E.g., if-stmt → **IF LPAREN expr RPAREN stmt**



terminal

CFG Example

A pure λ -term is defined inductively as follows:

- Any variable x is a λ -term
- If e is a λ -term, so is $\lambda x. e$ (abstraction)
- If e_1, e_2 are λ -terms, so is $e_1 e_2$ (application)

Add WeChat powcoder
term → var

| term term

| λ var . term

↑
terminals

Define Tokens via CFG

term → var

| term term
Assignment Project Exam Help

| λ var . term
<https://powcoder.com>

var → letter var

| Add WeChat powcoder
letter

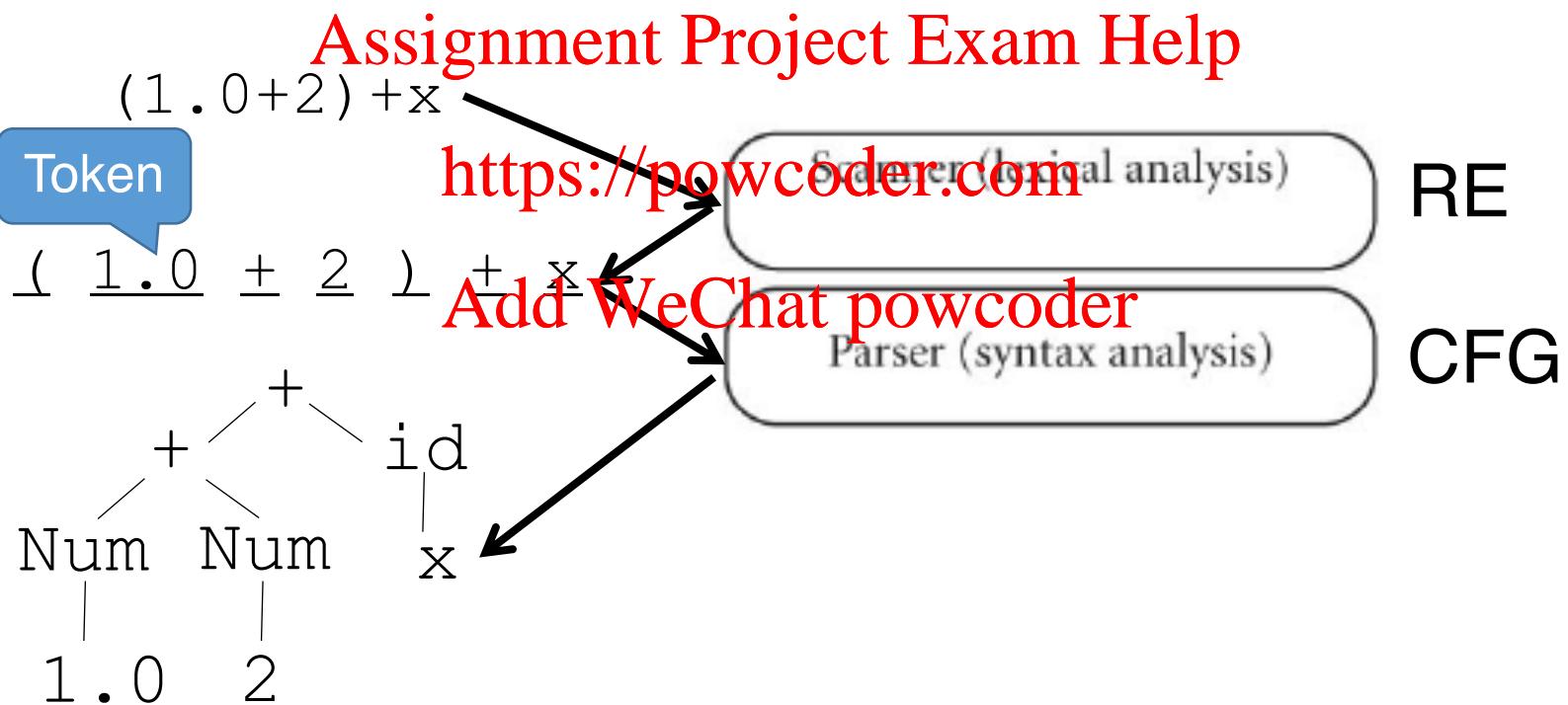
letter → a | b | ... | z

In RE: var → ([a – z] | [A – Z])⁺

CFG Example

expr → id | number | - expr
Assignment Project Exam Help
| (expr) | expr op expr
op → + | - | * | /
https://powcoder.com
Add WeChat powcoder
...

Typical Use of RE and CFG



Define Tokens by Scanner

expr → id | number | \neg expr
Assignment Project Exam Help
| (expr) | expr op expr
<https://powcoder.com>

Add WeChat powcoder
where id number op are tokens
recognized by the scanner

Build CFG

All strings with n 0's followed by n 1's, $n \geq 1$?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Build CFG

All strings with n 0's and n 1's, $n \geq 0$?

Assignment Project Exam Help

$S \rightarrow 0S1S \mid 1S0S \mid \epsilon$

Add WeChat powcoder

RE and CFG

CFG is strictly more expressive than RE

1. For any RE R , there is CFG C , such that $L(R) = L(C)$
2. Exists CFG C , such that for all RE R , $L(R) \neq L(C)$

<https://powcoder.com>

Add WeChat powcoder

1. Proof by induction.
2. A language with n 0's followed by n 1's, $n \geq 1$

$$S \rightarrow 0S1 \mid 01$$

RE

CFG

ϵ

$S \rightarrow \epsilon$

a

Assignment Project Exam Help

$R_1 R_2$

$S \rightarrow R_1 R_2$

$R_1 | R_2$

$S \rightarrow R_1 | R_2$

R^*

$S \rightarrow S R | \epsilon$

where R_1, R_2 and R are the equivalent nonterminal for R_1, R_2 and R in RE respectively

#15

Assignment Project Exam Help
Syntax <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Context-Free Grammar (CFG)

A CFG consists of

- A set of *terminals* T (basic alphabet)
Assignment Project Exam Help
<https://powcoder.com>
- A set of *non-terminals* N
- A *start symbol* S (a non-terminal)
- A set of *production rules*
Add WeChat powcoder

$$A \rightarrow \omega$$

nonterminal

string of terminals and nonterminals, separated by,
‘|’ meaning alternation
‘ ’ meaning concatenation

What strings are defined by a CFG?

Context-Free Language

$L(G)$: the language (set of strings) defined by G

Assignment Project Exam Help

Definition:

$L(G)$ is the set of all terminal strings **derivable** from the start symbol S

Derivation

A sequence from start symbol, each step a non-terminal is replaced by RHS of a production

Assignment Project Exam Help

A derivation of "xyz" with rule

$\text{var} \Rightarrow \text{letter var}$

$\Rightarrow \text{letter letter var} \Rightarrow \text{letter letter letter}$

$\Rightarrow x \text{ letter letter} \Rightarrow x y \text{ letter} \Rightarrow x y z$

Var	$\rightarrow \text{letter var}$
	$ \text{ letter}$
	$\text{letter} \rightarrow a b \dots z$

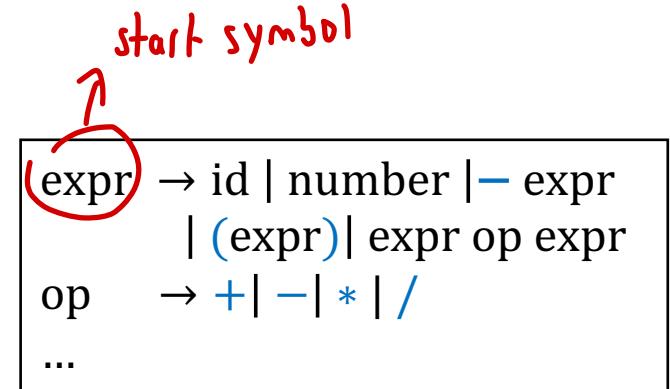
A multi-step reduction

$\text{var} \Rightarrow^* xyz$

Derivation

A derivation of “x+3*y” with rule

$\text{expr} \Rightarrow \text{expr op expr} \Rightarrow \text{id op expr} \Rightarrow \text{id op expr op expr}$
 $\Rightarrow \text{id op number op expr} \Rightarrow \text{id op number op id}$
 $\Rightarrow \text{x op number op id} \Rightarrow \text{x + number op id}$
 $\Rightarrow \dots \Rightarrow \text{x + 3 * y}$



Add WeChat powcoder

A reduction

$\text{expr} \Rightarrow^* \text{x + 3 * y}$

Order of Derivation

Derivation can follow any order

Assignment Project Exam Help
Leftmost derivation of "xyz"

var \Rightarrow letter var
 \Rightarrow x var \Rightarrow x letter var
 \Rightarrow x y var \Rightarrow Add WeChat powcoder

var	\rightarrow letter var
	letter
letter	\rightarrow a b ... Z

Rightmost derivation of "xyz"

var \Rightarrow letter var
 \Rightarrow letter letter var \Rightarrow letter letter letter
 \Rightarrow letter letter z \Rightarrow letter y z \Rightarrow x y z

$S \Rightarrow SS^+$
 $\quad \quad \quad \Rightarrow Sa^+$
 $\quad \quad \quad \Rightarrow S S^+ a^+$
 $\quad \quad \quad \Rightarrow S a^+ a^+$
 $\quad \quad \quad \Rightarrow S S^+ a^+ a^+$
 $\quad \quad \quad \Rightarrow S a^+ a^+ a^+$

Writing derivation can be tedious $\rightarrow a^+ a^+ a^+$ 6

Parse Tree

Derivation in graphical form

Assignment Project Exam Help

Root: the start symbol

<https://powcoder.com>

Leaf: terminal

parent

Add WeChat powcoder

child₁ child₂ ... child_n

represents

parent \Rightarrow child₁ child₂ ... child_n

Parse Tree

Derivation in graphical form



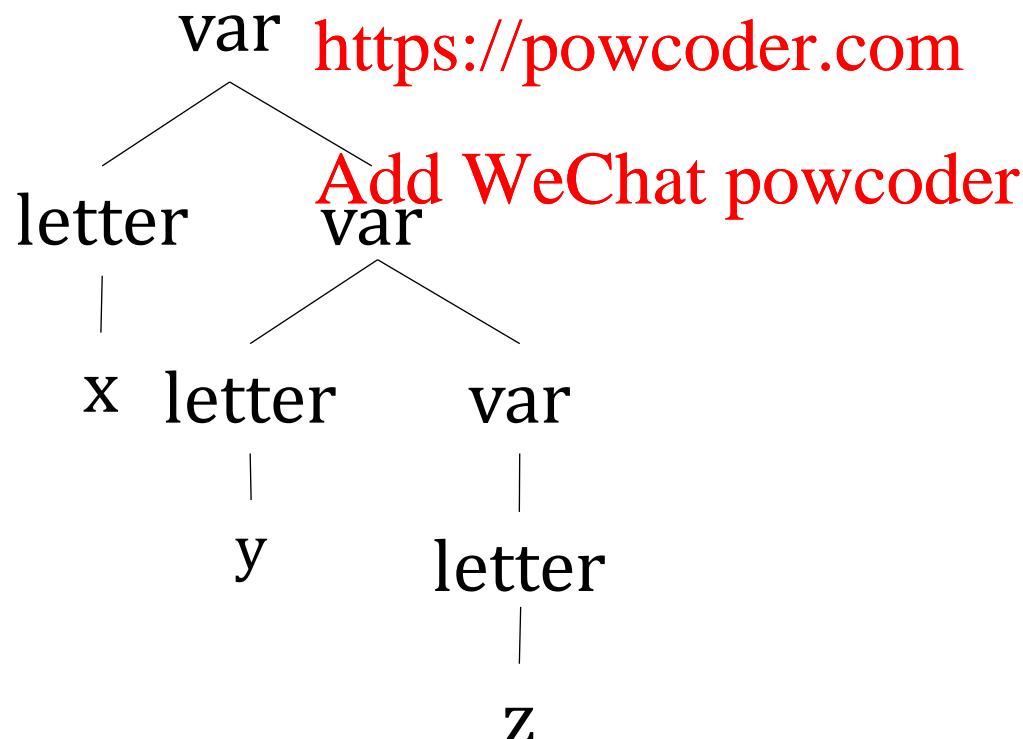
Parse Tree Example

Leftmost derivation of “xyz”

var \Rightarrow letter var

\Rightarrow x var \Rightarrow x letter var

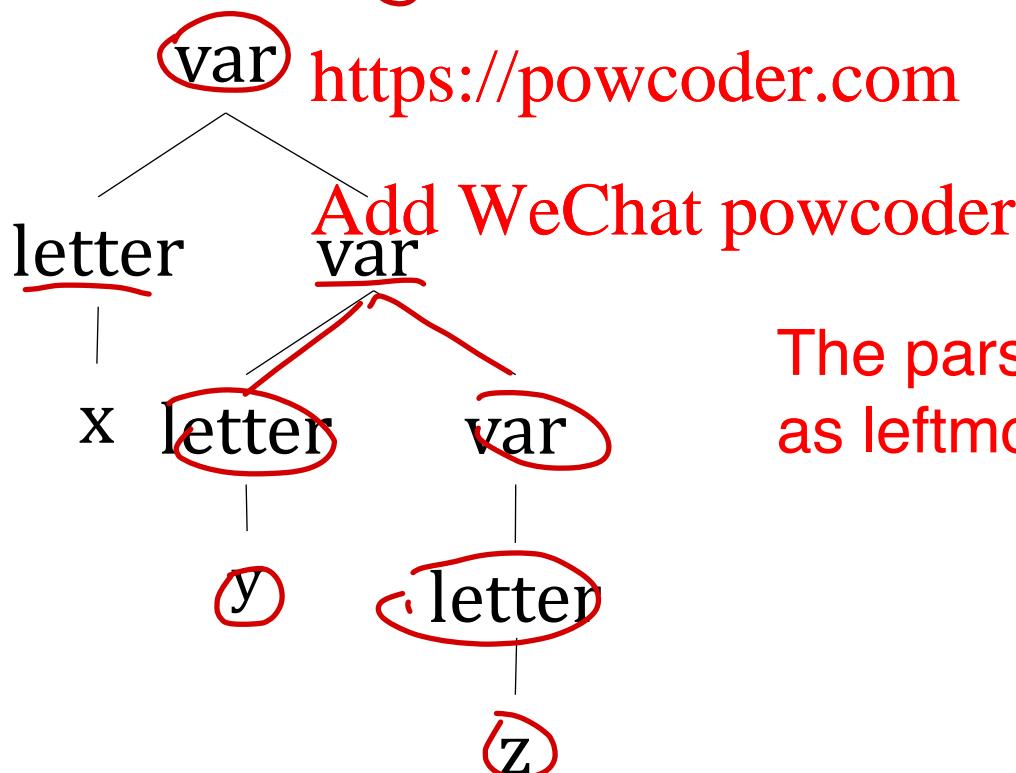
\Rightarrow xyz var \Rightarrow xyz letter \Rightarrow xyz



Parse Tree Example

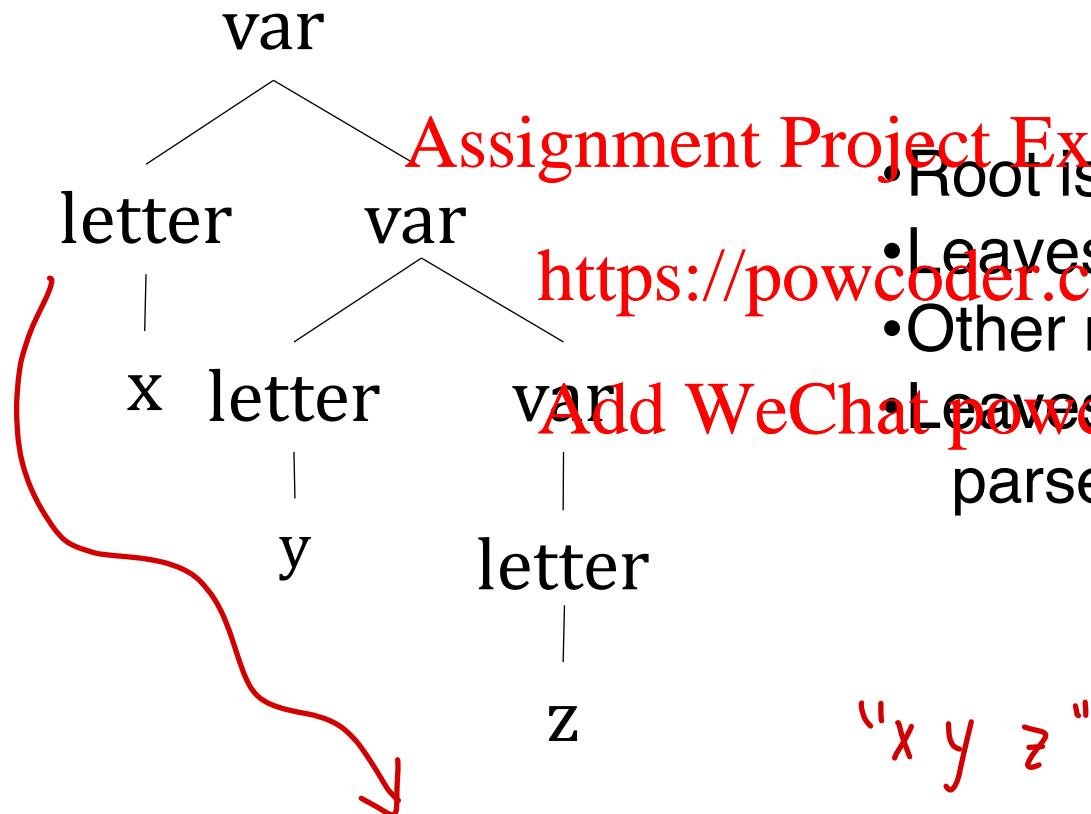
Rightmost derivation of “xyz”

~~var~~ \Rightarrow letter var
 \Rightarrow letter ~~letter~~ ~~var~~ \Rightarrow letter letter letter
 \Rightarrow letter letter ~~letter~~ ~~letter~~ ~~var~~ \Rightarrow xyz
Assignment Project Exam Help



Add WeChat powcoder
The parse tree is the same as leftmost derivation!

Parse Tree Features



Assignment Project Exam Help

• Root is the start symbol

• Leaves are terminals

• Other nodes are nonterminals

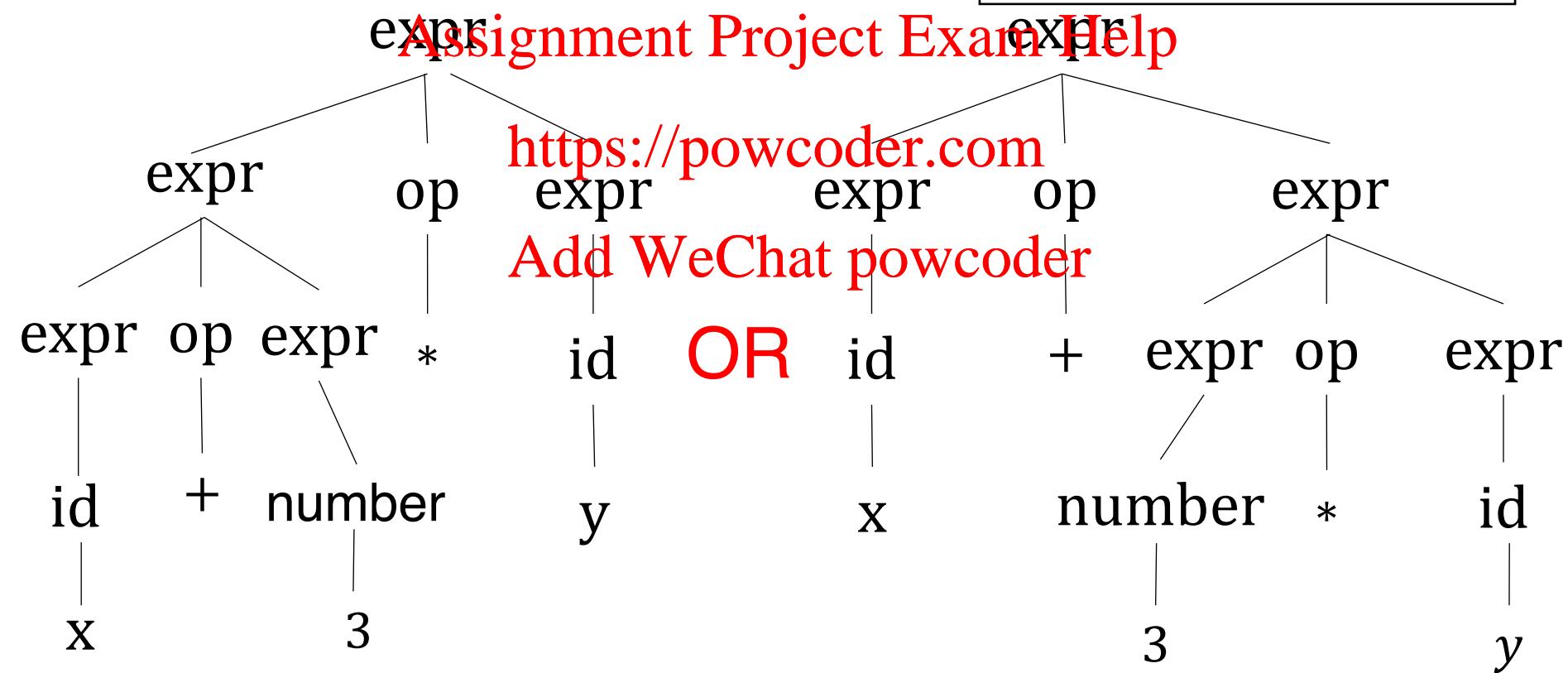
• Leaves (left to right) form the parsed string

"x y z"

Parse Tree Example

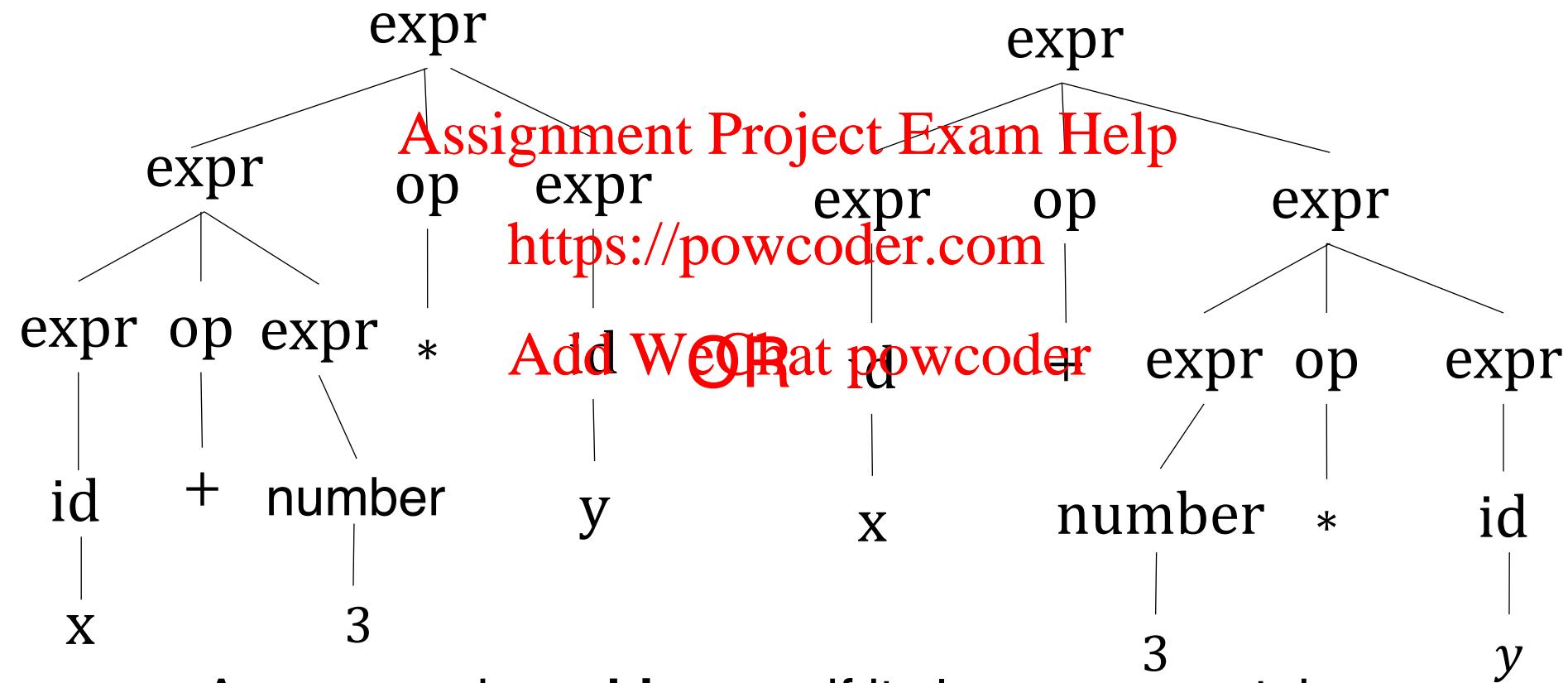
parse trees for “x+3*y”

```
expr → id | number | - expr  
      | (expr) | expr op expr  
op   → + | - | * | /  
...
```



Ambiguity

Two different parse trees for “x+3*y”



A grammar is **ambiguous** if its language contains at least one string with two or more parse trees

Ambiguity

Is the following language ambiguous?

Assignment Project Exam Help

$$\begin{aligned} \text{expr} &\rightarrow \text{id} \mid \text{number} \mid - \text{expr} \\ &\quad \mid \frac{\text{https://powcoder.com}}{(\text{expr})} \mid \text{expr op expr} \\ \text{op} &\rightarrow + \mid - \mid * \mid / \end{aligned}$$

Yes, two parse trees (previous slide) for string $x+3^*y$

Expression Grammar

Precedence: which operator should be evaluated sooner

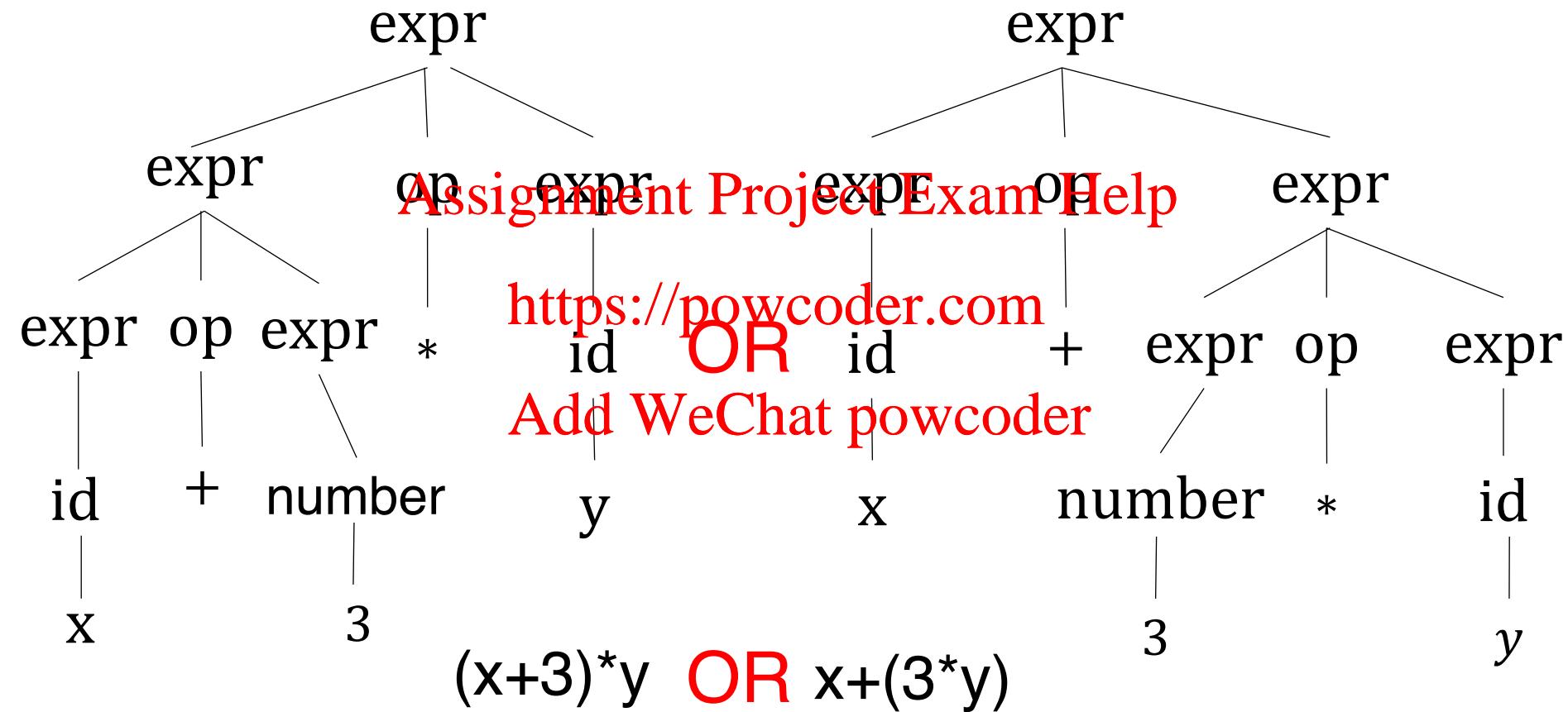
Assignment Project Exam Help

Arithmetic: * and / have higher precedence;
all operators are left-associative

Add WeChat powcoder

Associativity: (binary) operator with same precedence evaluated left-to-right or right-to-left

Precedence



The lower an operation is, the higher precedence it has

Defining Precedence in Grammar

Define operations at different “levels”

expr → id | number | – expr
Assignment Project Exam Help
| (expr) | expr op expr
op → + | - | * | / | ,
<https://powcoder.com>

Add WeChat powcoder

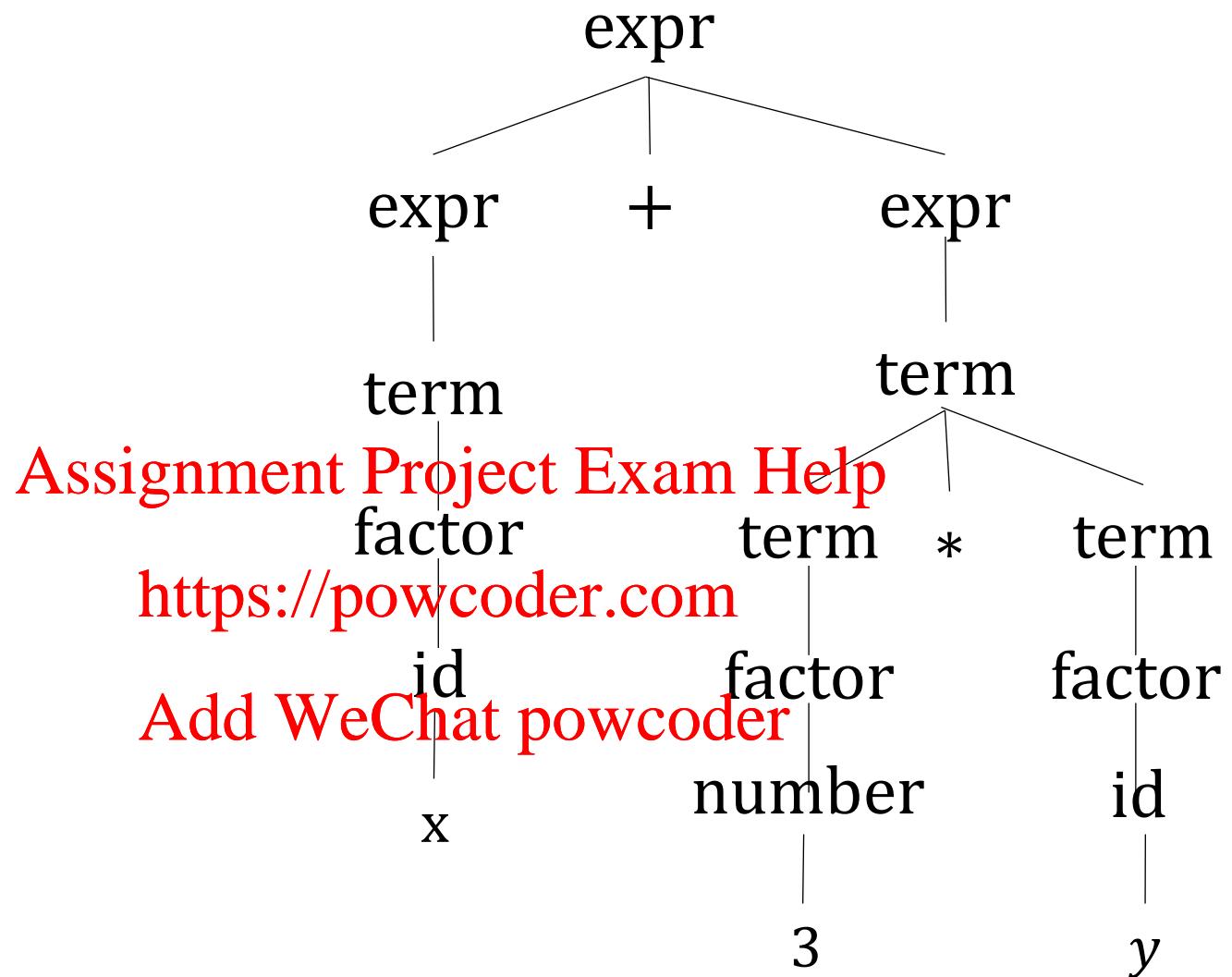

expr → expr + expr | expr – expr | term
term → term * term | term / term | factor
factor → id | number | (expr) | – factor

Level1

Level2

Level3

The farther from start symbol, the higher precedence



$x + 3^*y$

Only one parse tree for $x + 3^*y$

#16

Assignment Project Exam Help

Syntax <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Defining Precedence in Grammar

Define operations at different “levels”

expr → id | number | – expr
Assignment Project Exam Help
| (expr) | expr op expr
op → + | - | * | / | ,
<https://powcoder.com>

Add WeChat powcoder


expr → expr + expr | expr – expr | term
term → term * term | term / term | factor
factor → id | number | (expr) | – factor

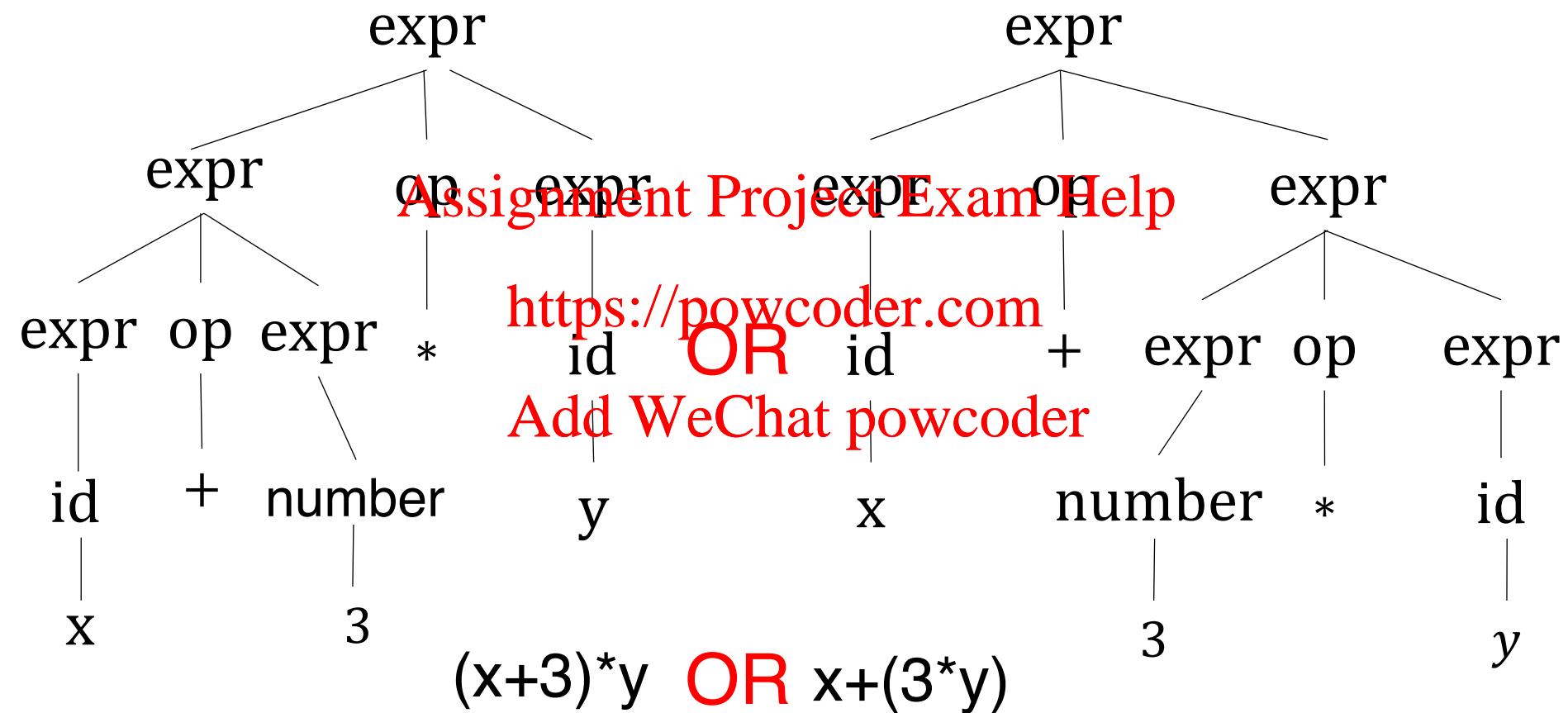
Level1

Level2

Level3

The farther from start symbol, the higher precedence

Ambiguity



String with different parse tree
might have different meanings

Associativity of Binary Operators

A property of *binary operations*

... + a + : sign “+” is left-associative since
a is associated with the left “+”
<https://powcoder.com>

An operator AddWeChat powcoder associativity
is evaluated left-to-right (right-to-left)

Left: +,-,*,/

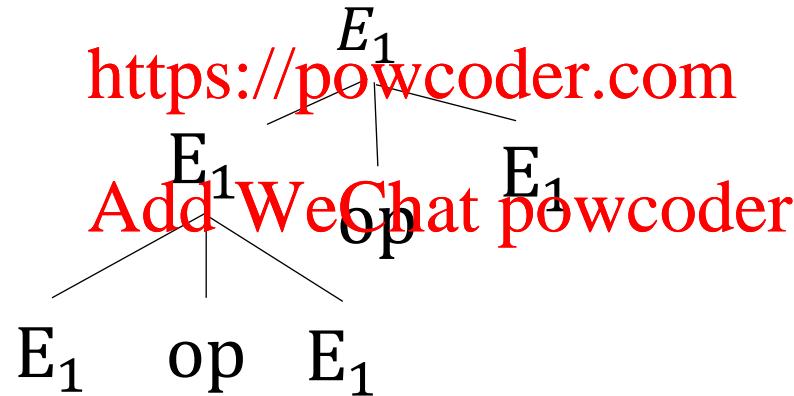
Right: = in C (a=b=c same as a=(b=c))

Associativity in CFG

Left-recursion: LHS is the start of RHS in a derivation

$$E_1 \xrightarrow{*} E_1 \text{ op } \dots$$

Defines left-associativity on op



Right-recursion: LHS is the end of RHS in a derivation

$$E_1 \xrightarrow{*} \dots \text{ op } E_1$$

Defines right-associativity on op

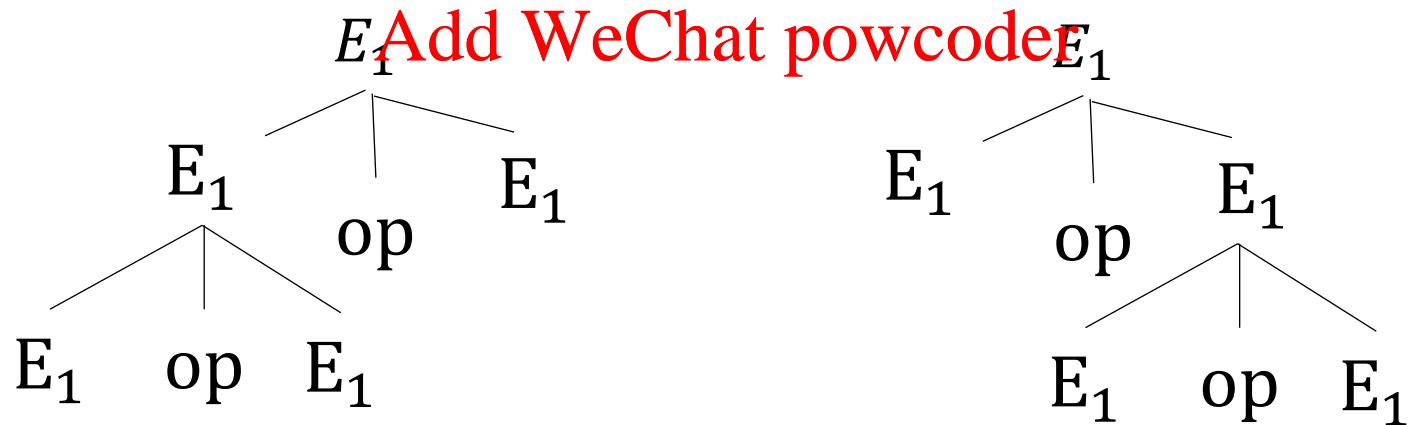
Recursion and Ambiguity

Left-recursion: $E_1 \rightarrow^* E_1 \text{ op } \dots$

Right-recursion: $E_1 \rightarrow^* \dots \text{ op } E_1$

Assignment Project Exam Help

A grammar with at least one operation that is both left and right recursive is ambiguous
<https://powcoder.com>



Two parse trees for $E_1 \text{ op } E_1 \text{ op } E_1$

Direct Recursion in Grammar

Left-recursion: $E_1 \rightarrow^* E_1 \text{ op } \dots$

Right-recursion: $E_1 \rightarrow^* \dots \text{ op } E_1$

Assignment Project Exam Help

expr \rightarrow exp~~https://powcoder.com~~ | term

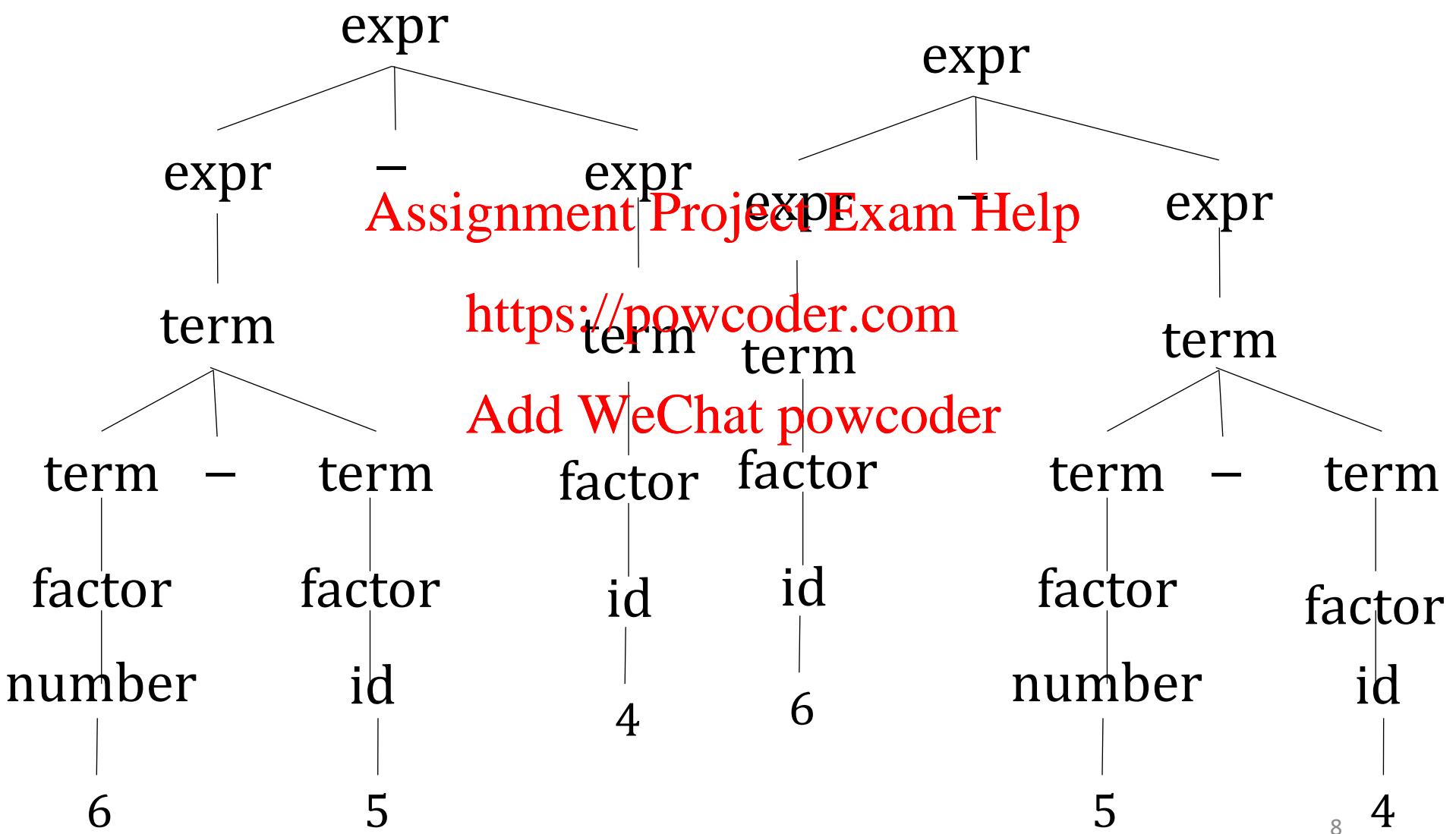
term \rightarrow term^{*} term | term/term | factor

factor \rightarrow id | number | (expr) | – factor

The production rule of $+, -, *, /$ is both left- and right-recursive. So the grammar is ambiguous.

Different parse trees for string “6-5-4”

Left tree evaluates to -3; right tree evaluates to 5



Indirect Recursion in Grammar

Left-recursion: $E_1 \rightarrow^* E_1 \text{ op } \dots$

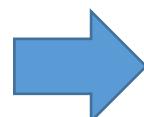
Right-recursion: $E_1 \rightarrow^* \dots \text{ op } E_1$

Assignment Project Exam Help

equals → var = equals1

equals1 → Add WeChat | var

Indirect recursion: equals →^{*} var = equals



= is
right-associative

Defining Associativity in Grammar

expr → expr + expr | expr – expr | term
term → term * term | term / term | factor
factor

~~Assignment Project Exam Help~~

<https://powcoder.com>

Add WeChat [powcoder](https://powcoder.com)

Remove right-recursion

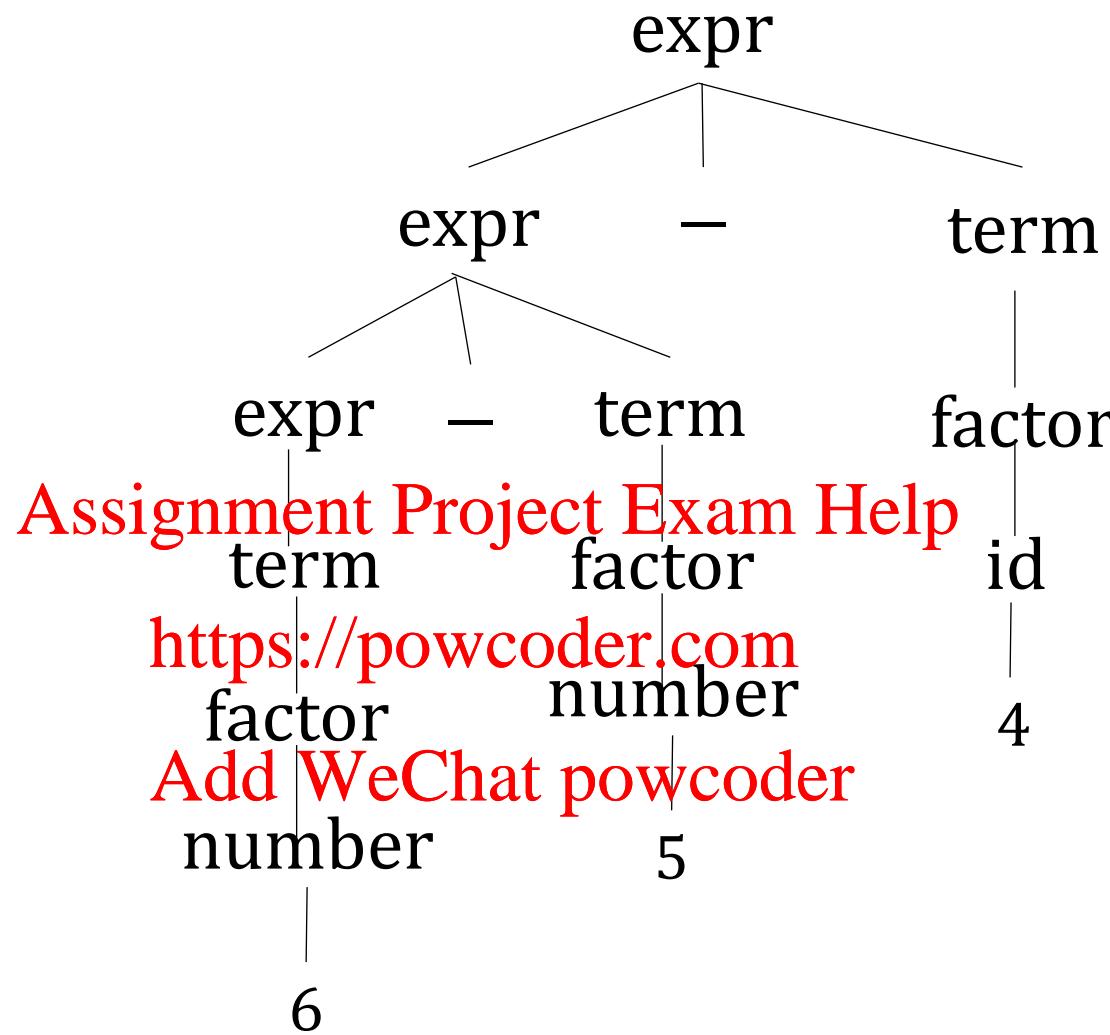
on +, -, *, /

expr → expr + term | expr – term | term

term → term * factor | term / factor | factor

factor → id | number | (expr) | – factor

In this grammar: +, –, *, / are left-associative



$(6-5)-4$

Only one parse tree for $6-5-4$

Dangling Else Ambiguity

if (b1)

if (b2)
else c2

stmt

if (expr) stmt

if (expr) stmt
else stmt

if (b1)

if (b2) c1
else c2

stmt

if (expr) stmt else

if (expr) stmt

Add WeChat powcoder

OR

stmt

Grammar: $\text{stmt} \rightarrow \text{if}(\text{expr}) \text{stmt} \mid \text{if}(\text{expr}) \text{stmt} \text{ else} \text{stmt}$

Avoid Dangling Else

Approach 1: Change syntax (ALGOL, Ada)

Grammar: stmt → if (expr) stmt fi | if (expr) stmt else stmt fi

Assignment Project Exam Help

```
if (b1)      https://powcoder.com(b1)
  if (b2)  c1      if (b2)  c1
    else  c2          fi
  fi
else  c2
fi
```

Avoid Dangling Else

Approach 2: Keep syntax, change grammar

Grammar: stmt → matched | unmatched

matched → if (expr) matched else matched

unmatched → if (expr) stmt
<https://powcoder.com>

| if (expr) matched else unmatched

Add WeChat powcoder

if (b1)

 if (b2) c1

 else c2



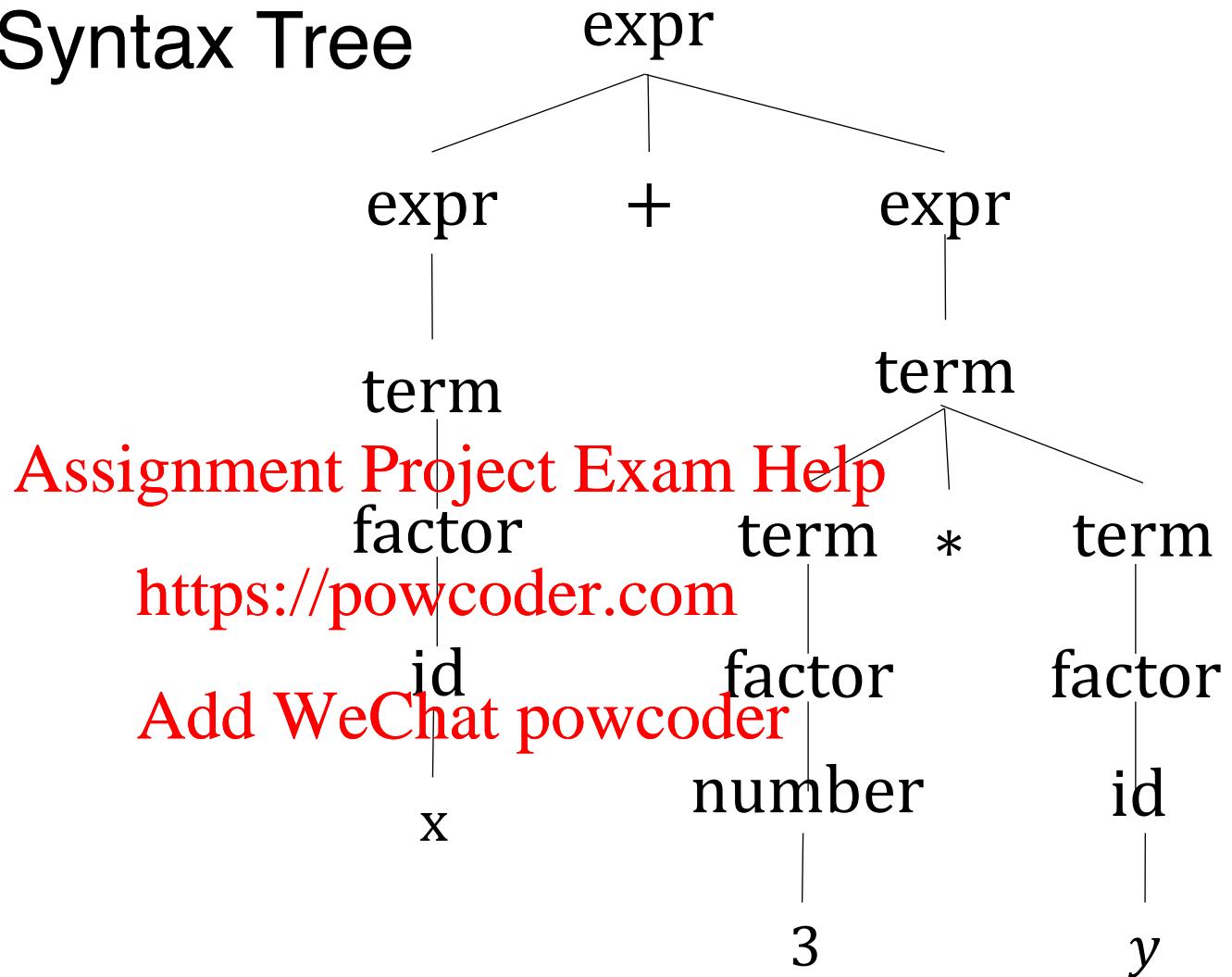
if (b1)

 if (b2) c1

 else c2

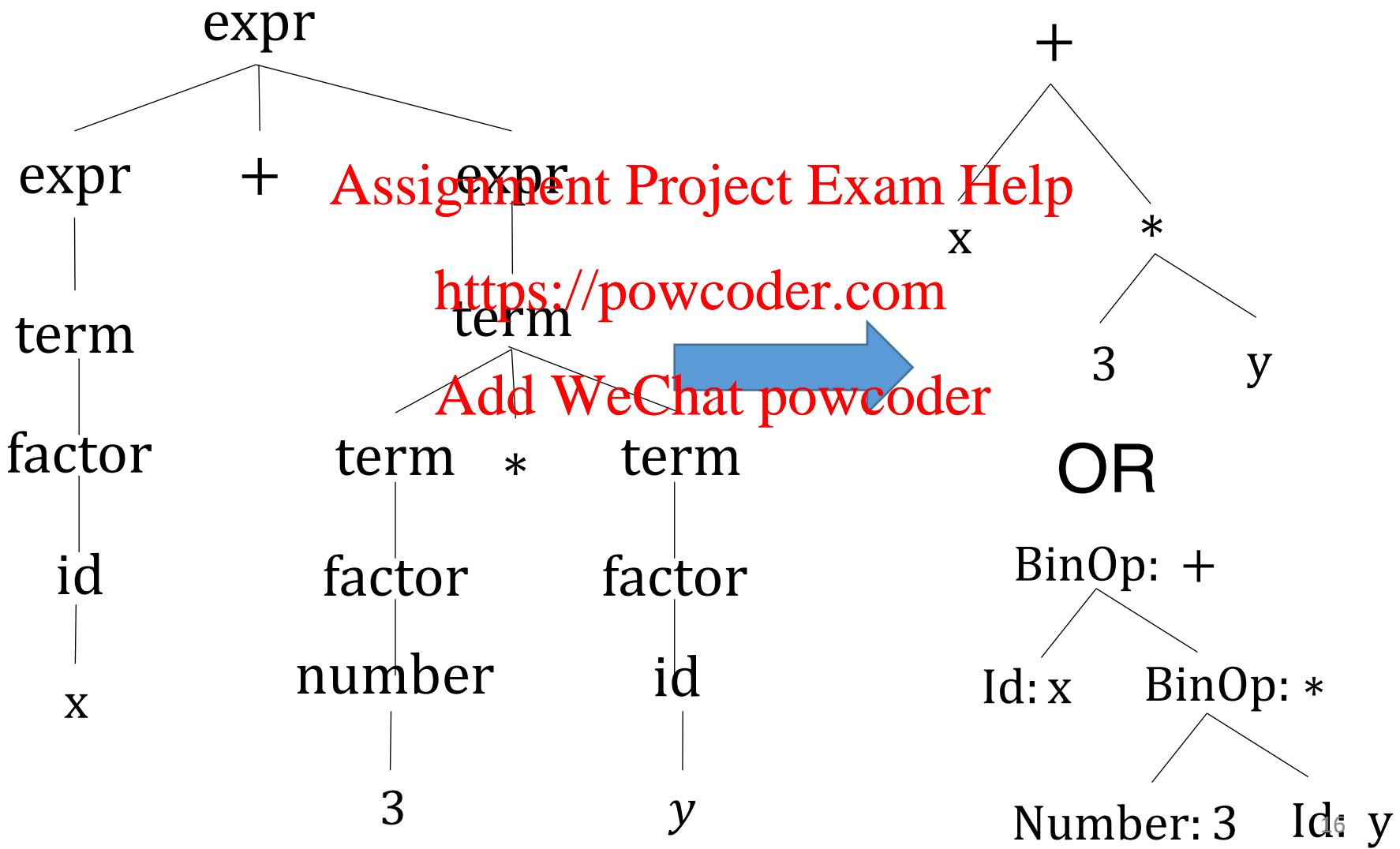


Concrete Syntax Tree

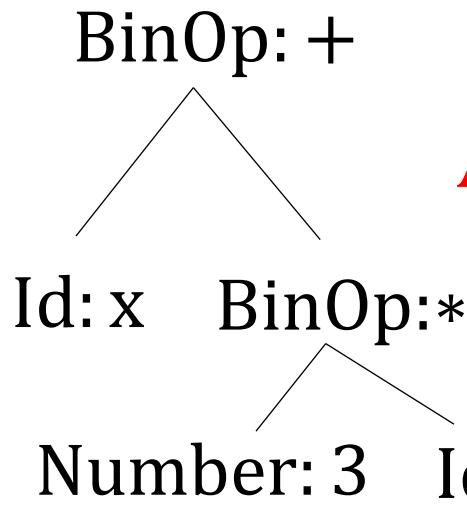


The parse tree is very verbose. It tracks information only useful for parsing (e.g., defining precedence and associativity)

Abstract Syntax Tree (AST)



Abstract Syntax Tree (AST)



AST doesn't show the whole syntactic clutter,
e.g., parentheses, nonterminals added for unambiguity

Assignment Project Exam Help

AST keeps the same structure as parse tree
<https://powcoder.com>

Each node usually corresponds to an object in impl.
Add WeChat powcoder
(more manageable for later compiler stages)

AST hides syntactical language differences

Scheme: (* 3 y)
BinOp: *

Number: 3 Id: y

C, Java: (3 * y)
BinOp: *

Number: 3 Id: y

Expression Gramma

- Define precedence (grammar with different levels of operators)
- Define associativity (left-recursive vs. right-recursive derivations) to avoid ambiguity

Add WeChat powcoder

Avoiding ambiguity in general is undecidable

Abstract Syntax Tree (AST) is a compact representation of parsing tree

Parsing

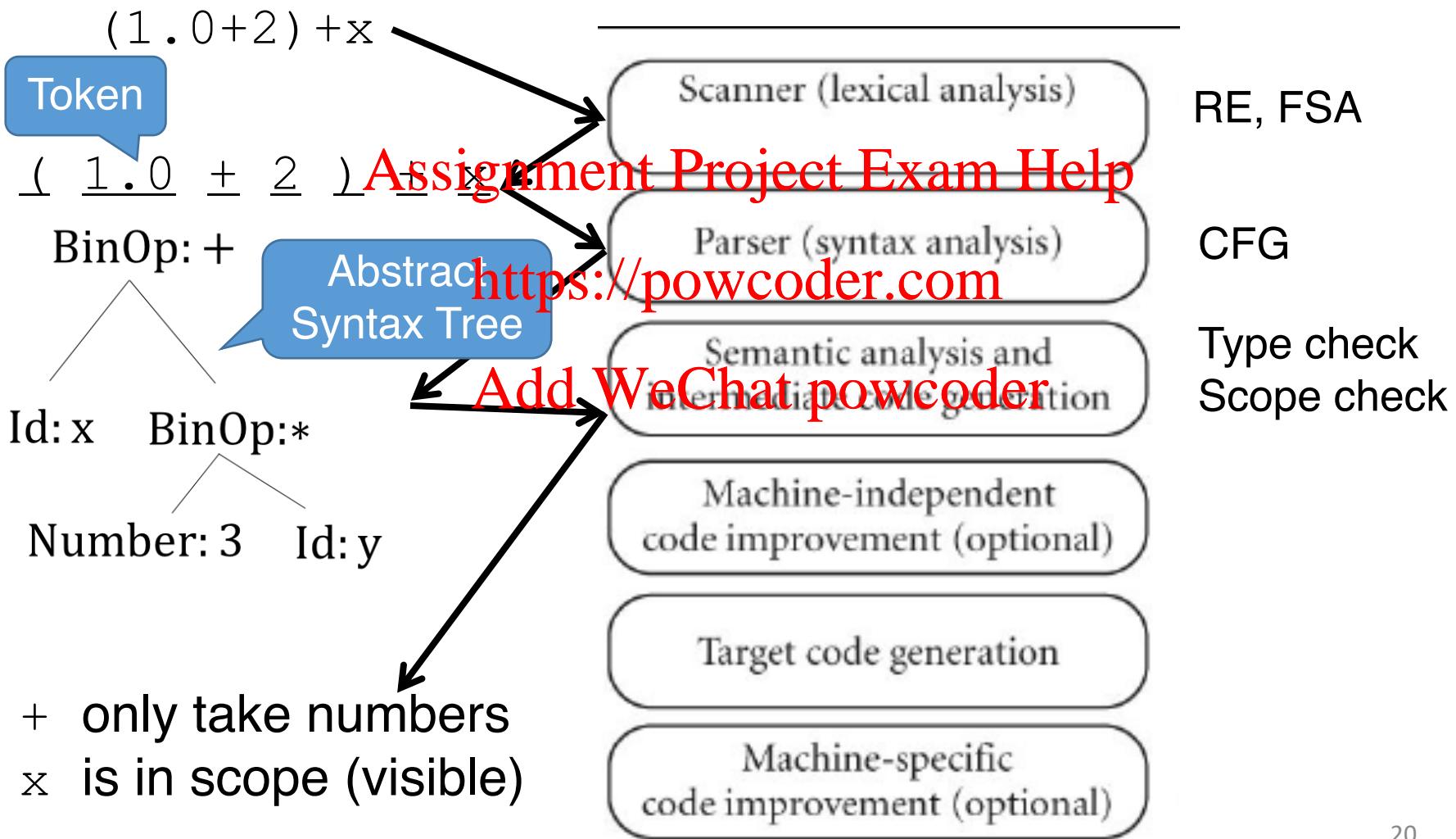
Assemble tokens (from scanner) to a syntax tree,
according to a CFG

Assignment Project Exam Help

In practice, most programming languages falls in
CFG with linear time parser (e.g., LL, LR)

Add WeChat powcoder

Parser generator (e.g., Yacc, Bison)
automatically generates parser from CFG



#17

Assignment Project Exam Help
Names, Scopes, Bindings
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Plan

HW3 due this Friday midnight

Assignment Project Exam Help

1st Mideterm covers materials from lecture 1 to 16, i.e.,
HW1 to HW3

<https://powcoder.com>
Add WeChat powcoder

Next Monday: review of assignments and practice problems

Next Wednesday: evening exam (no lecture)

Midterm 1

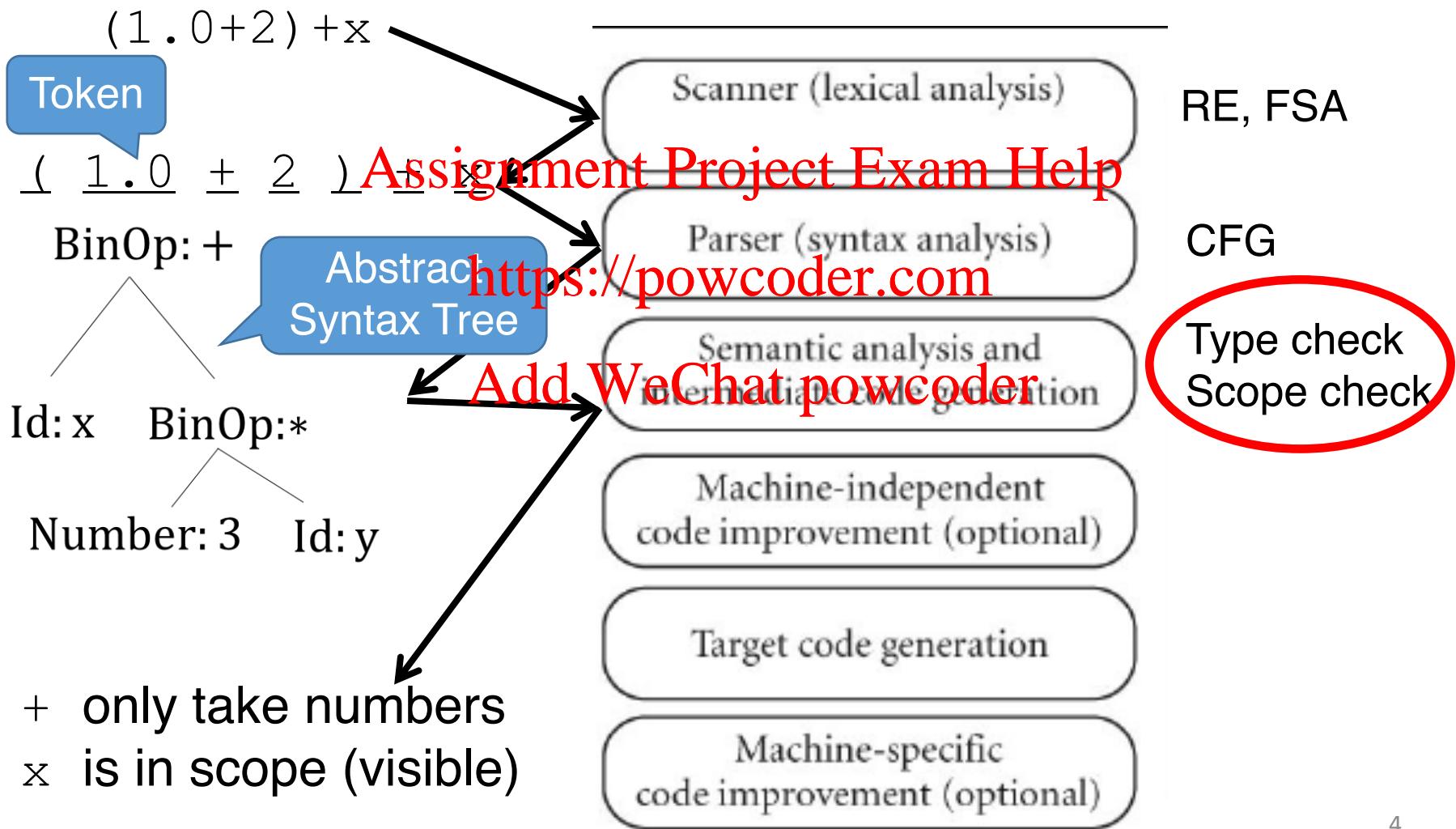
Carefully read the exam instructions (see Canvas announcement) before the exam

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Practice questions are posted on Gradescope (will not be graded)

<https://powcoder.com>
[Add WeChat powcoder](#)

Conflict exam: send me (dbz5017@psu.edu) and Zeyu (dxd437@psu.edu) with supporting documents, if any, to request a conflict exam *by this Friday*



Limitations of CFG

Grammar for a simple language:

```
Prog ::= Decl* Stmt*
Decl ::= INT ID, | bool ID;
Stmt ::= ID = Exp
```

Need Context Sensitive Information

Add WeChat powcoder

```
int x;  
y = 0
```

```
bool b;  
b = 0
```

y defined?

b number?

We need to add *meanings* to symbols when go beyond syntactic analysis

Names in Program

A mnemonic string in high-level languages

Assignment Project Exam Help

Identifiers in most languages

<https://powcoder.com>

An abstraction of low-level representation, such as memory address and register

Variables

A ***binding*** of a name to a memory address
(binding: symbol to object)

Assignment Project Exam Help

A variable usually has:

name, address, type, value, lifetime, scope

Add WeChat powcoder

Names in Programs

Scope: visibility of names
in source program

Static
Assignment Project Exam Help

```
seed = DateTime.Now.Millisecond;
number = new Random(seed);
int maskPos = number.Next(minPos,maxPos);
int j=0;
do
{
    seed = DateTime.Now.Millisecond;
    number = new Random(seed);
    int newPos = number.Next(1,9);
    int x = _setRowPosition[i]+p;
    int y = _setColPosition[i];
    if(_problemSet[x,y]==0)
        [x,y] = new
```

<https://powcoder.com>

Add WeChat powcoder

Storage: memory space
associated with names



Dynamic

Lifetime: the time interval a variable is allocated
with memory

Objects and Lifetime

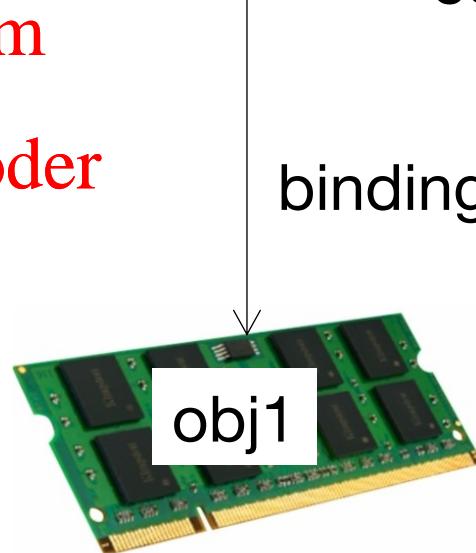
Run-time representation of a name

Key events of objects:

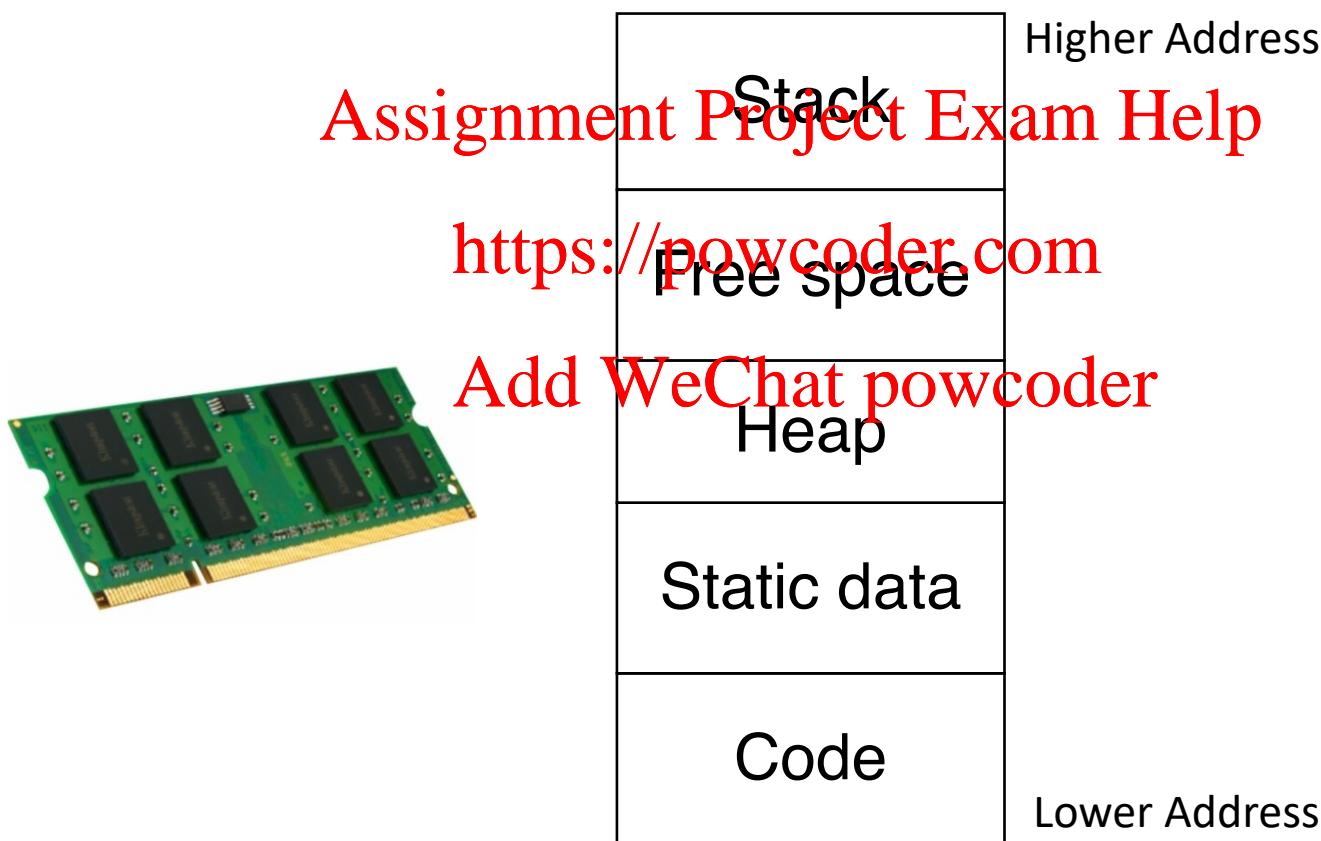
Assignment Project Exam Help

- Creation of objects
- References to variables
- (temporary) deactivation of bindings
- Reactivation of bindings
- Destruction of bindings
- Destruction of objects

source
code



Typical Memory Layout



Storage Allocation Mechanism and Object Lifetime

Static allocation: bound to memory cells before execution begins and remains bound to the same memory cell throughout execution

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

Stack allocation: storage bindings are created for variables when their declaration statements are elaborated

[Add WeChat](#) [powcoder](#)

Heap allocation: Nameless memory cells that are allocated and deallocated by explicit directives “run-time instructions”, specified by the programmer (either explicitly or implicitly)

Static Allocation

Lifetime of Objects: throughout execution

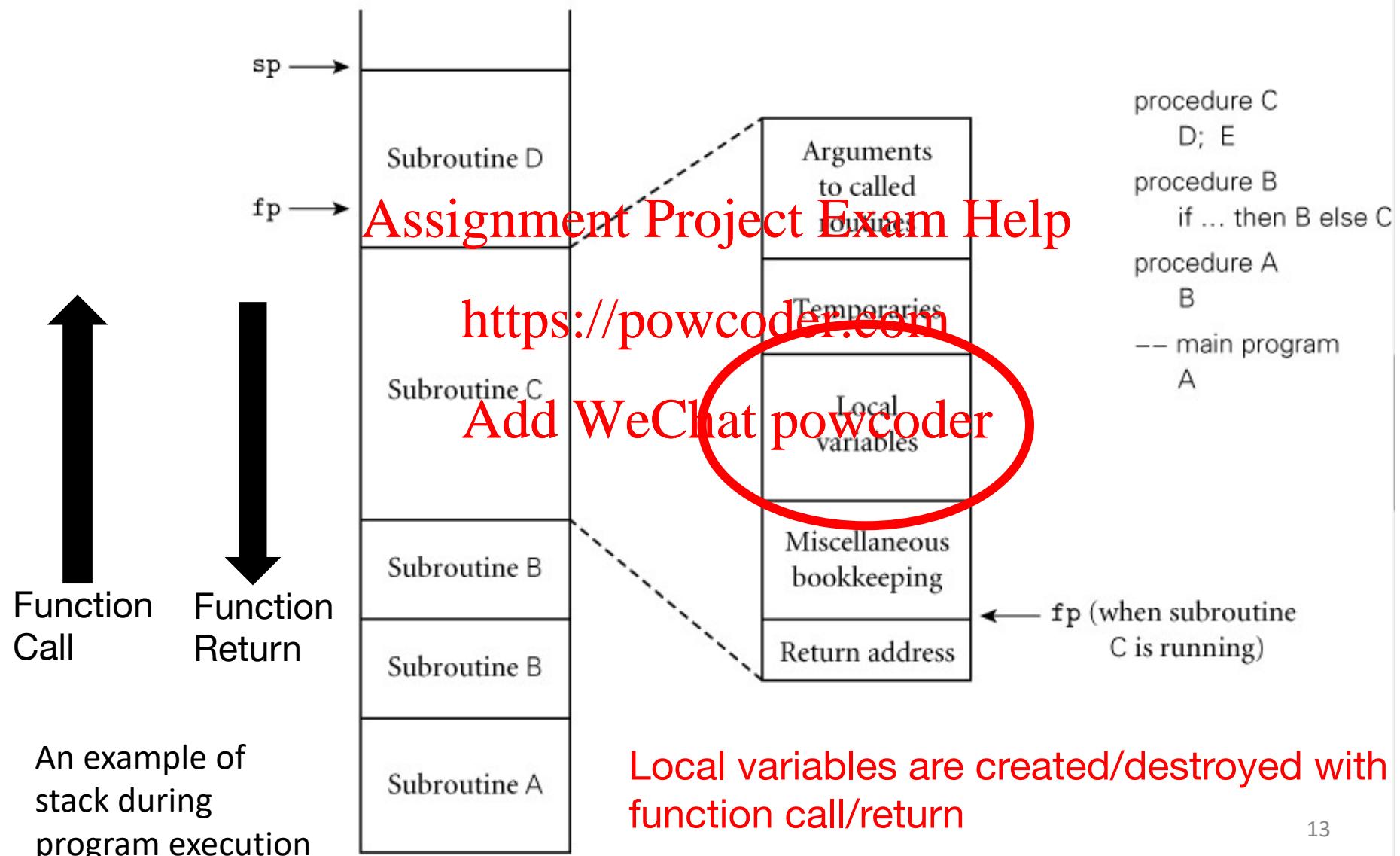
Assignment Project Exam Help

E.g., global variables, constants

<https://powcoder.com>

Objects allocated in static data area

Stack Allocation



Lifetime of Local Variables

When does the lifetime of each variable begin & end?

Assignment Project Exam Help

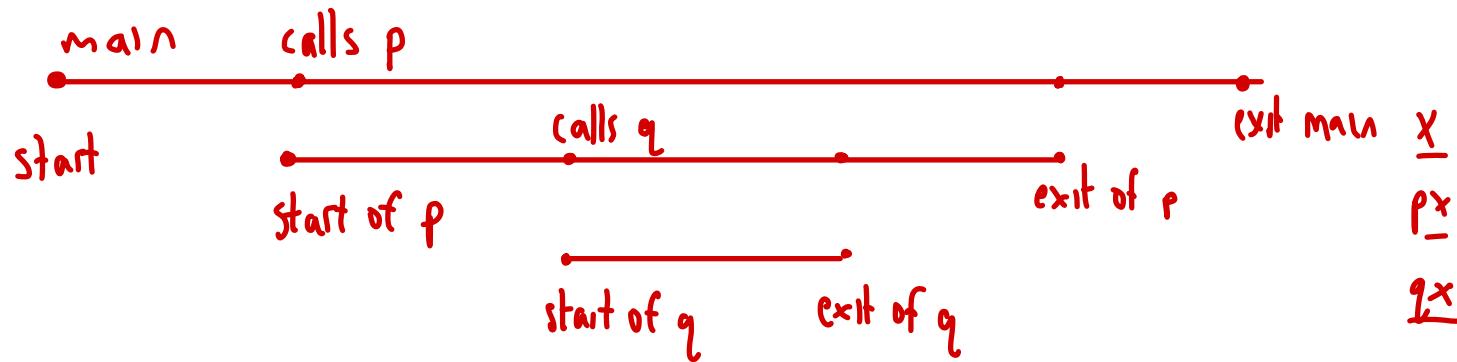
```
int main () {  
    int x;  
    p(x);  
    q(x, x);  
}
```

```
int p(int p1) {  
    int px;  
    q(p1, px);  
}
```

```
int q(int q1, q2) {  
    int qx;  
    ...  
}
```

<https://powcoder.com>

Add WeChat powcoder



Heap Allocation

Allocation Request (with size)

Assignment Project Exam Help
Free request <https://powcoder.com>



An example of heap during program execution

Typically used for user-requested storage

Lifetime of User-Requested Storage

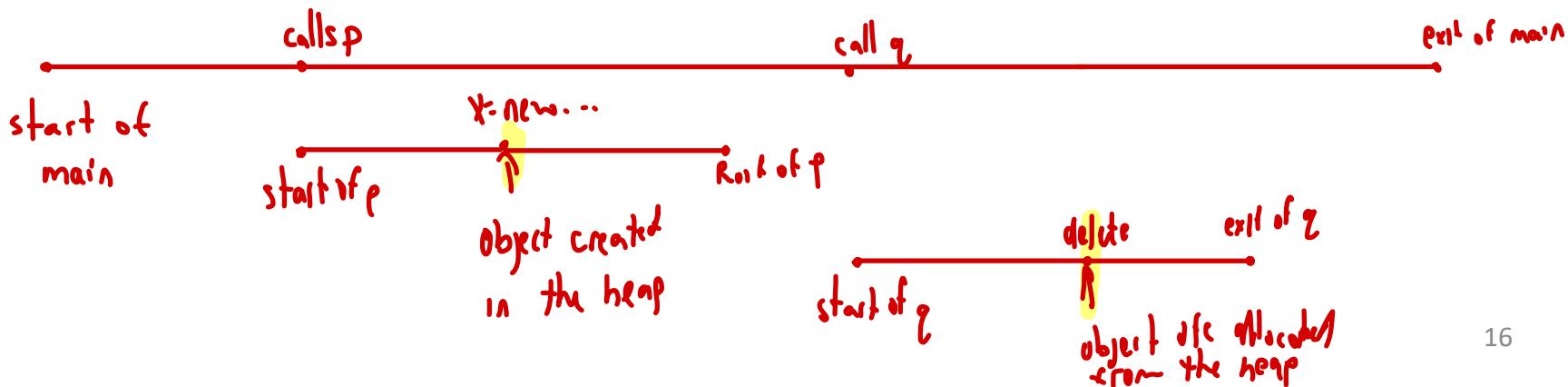
When does the lifetime of dynamic storage begin & end?

Assignment Project Exam Help

```
String* x;  
int main () {  
    p();  
    q();  
}
```

```
int p() {  
    ...  
    x = new String[10];  
}  
  
Add WeChat powcoder
```

```
int q() {  
    ...  
    delete x  
    ...  
}
```



#18

Assignment Project Exam Help
Names, Scopes, Bindings
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Names in Programs

Scope: visibility of names
in source program

Static

Assignment Project Exam Help

```
seed = DateTime.Now.Millisecond;
number = new Random(seed);
int maskPos = number.Next(minPos,maxPos);
int j=0;
do
{
    seed = DateTime.Now.Millisecond;
    number = new Random(seed);
    int newPos = number.Next(1,9);
    int x = _setRowPosition[i]+p;
    int y = _setColPosition[i];
    if(_problemSet[x,y]==0)
        [x,y] = newPos;
}
```

<https://powcoder.com>

Add WeChat powcoder

Storage: memory space
associated with names



Dynamic

Lifetime: the time interval a variable is allocated
with memory

Scope

The visibility or availability of names

- Which code region is a name visible?
[Assignment](#) [Project](#) [Exam](#) [Help](#)

When is it determined?
<https://powcoder.com>

- Static or Dynamic
[Add WeChat](#) [powcoder](#)
- Lexical or Control-Flow?

How & when does it change?

Static (Lexical) Scoping

Scopes can be determined by the compiler

Assignment Project Exam Help

All bindings for identifiers can be resolved by
examining the program
<https://powcoder.com>

Add WeChat powcoder

Typically, use the closest binding

Used in most languages (C, Java, Scheme)

(let* (name def)
.....
.....
(use names))

Scope Rules in Scheme

(let ((x 0))
 (+ x x)) Scope of x

name
(let*) Scope of x
(let* ((x 0) (y x))
 (+ x y))

name
Assignment Project Exam Help
(let ((x 0))
 (let ((y (+ x 1)))
 (+ x y)))

Scope of inner y
(let ((x 0) (y 1))
 (let ((y (+ x 1))
 (x (+ y 1))))
 (+ x y)))

Add WeChat powcoder which y?

(define (f x)
 (g x x)) Scope of x

name
(define (f x)
 (let ((g *))
 (g x x))) Scope of g

Scope of z

(let ((x 17))
 (lambda (z) (+ z x)))
 name

(define (f x)
 (let ((*))
 (* x x))) Scope of *

Nested Scope

In Scheme, each (let ...) defines one scope

```
(let ((x 0))
```

Assignment Project Exam Help

In Pascal, ALGOL, function can be nested
<https://powcoder.com>

```
function E (int x) {  
    function F (int y)  
    return F(1)+x; }  
// F is not visible here
```

In C, block scopes can be nested

```
function E (int x) {  
    { int y=3; x=x+y; }  
    // y is not visible here  
    return x; }
```

How are
scope rules
being checked?

Scope Check

Symbol table: a table of names and their bindings

Assignment Project Exam Help

Single scope: a dictionary or hash map
<https://powcoder.com>

Nested scope: a tree of symbol tables

- Each scope has one symbol table
- Each symbol table may have a parent

Symbol Table

Each entry in the symbol table contains

- The name of an identifier
- Additional information: its kind, its type, if it is constant and so on (compiler dependent)

Add WeChat powcoder

Name	Kind	Type	Constant?
x	id	int	0

Scope Check Example

each scope has one symbol table

```
(let ((x 0) (y 7))  
  (let ((y (+ x 1))  
        (x (+ y 1)))  
    (+ x y)))
```

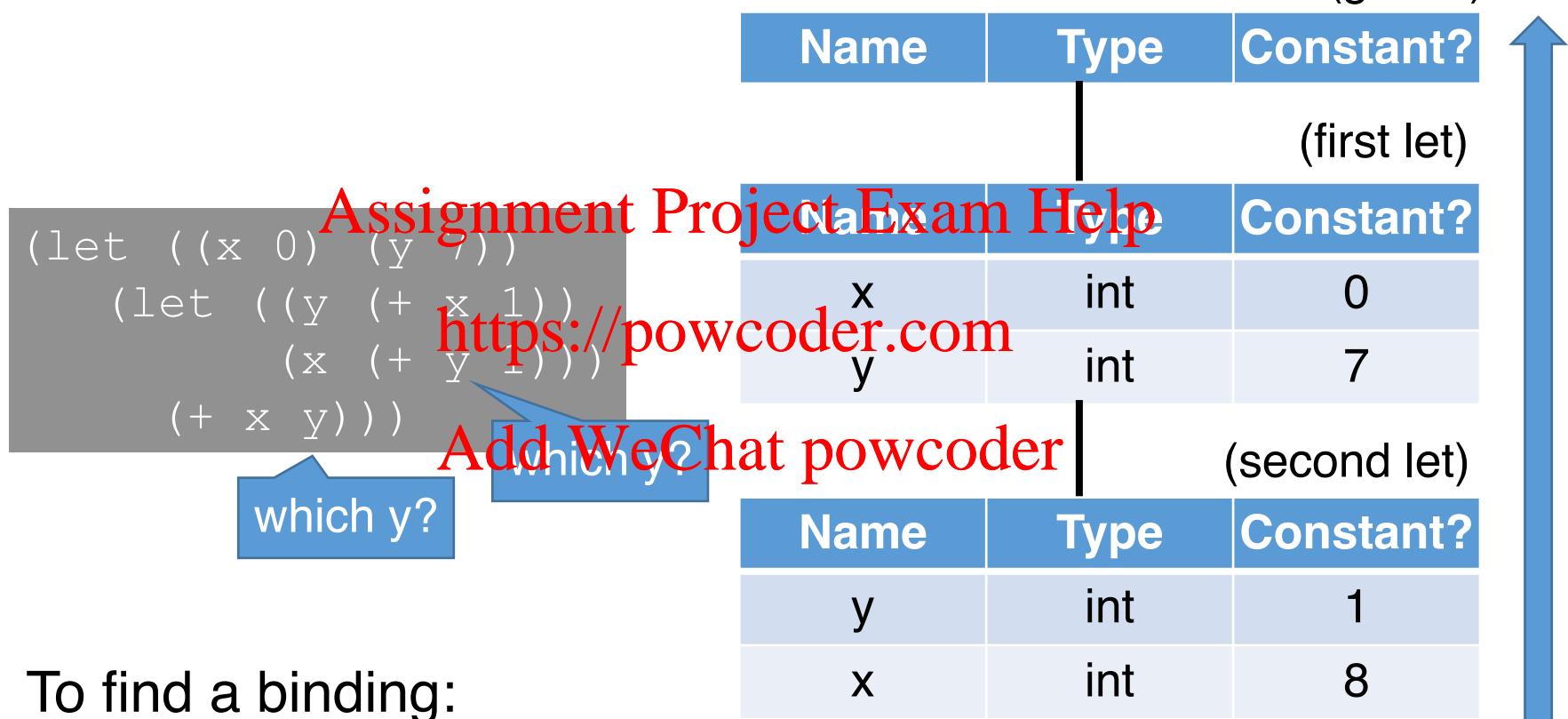
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Name	Type	Constant?
		(global)
		(first let)
Name	Type	Constant?
x	int	0
y	int	7
		(second let)
Name	Type	Constant?
y	int	1
x	int	8

Name Collision: Identifier with Same Name



To find a binding:

1. Start from table of current scope
2. Go up until the name is found
3. Fail if no table contains the name

Name
Search

Scope Check Example

```
int x;  
  
void f(int m) {  
    float x, y; In scope?  
    ...  
    { int i, j; x = 1; } In scope?  
    { int x; l; x = 2; }  
}
```

```
int g(int n) {  
    bool t;  
    x = 3; In scope?  
}
```

Global symtab

x	var	int
f	fun	int → void
g	fun	int → int

Assignment Project Exam Help
<https://powcoder.com>

func f
symtab

m	par	int
x	var	float
y	var	float

Add WeChat powcoder

func g
symtab

n	par	int
t	var	bool

i	var	int
j	var	int

x	var	int
l	lab	

CMPSC 461: Programming Language Concepts

Midterm 1 Practice Questions

Problem 1 λ -calculus.

- What are the free variables in the λ -term $(\lambda x . (\lambda y . y z) y) x$.
- Fully evaluate $(\lambda x . (\lambda y . ((\lambda y . y) x))) z$.
- Define a function `doubleChurch`, which takes a Church Numeral and doubles it. Try to build it from scratch, without using the encoding of `+` or `*` as defined in lecture.

Problem 2 Higher-order functions and currying.

- What is the curried form of $\lambda x y . (+ x y)$.
- Explain in one sentence, what's the result of $(\lambda x y . (+ x y)) 1$.

Problem 3 Scheme.

- Define a recursive Scheme function that takes a list of numbers and returns the product of all the list elements. Given the empty list, the product should be 1.
- Define the same functions using “`foldr`”.

Assignment Project Exam Help

Problem 4 Scheme.

<https://powcoder.com>

What do the following functions do?

```
(define (blue p l) (= (length l) (length (filter p l))))
(define (remap f x ys) (map (lambda (y) (f x y)) ys))
(define (binmap f xs ys) (map (lambda (x) (remap f x ys)) xs))
```

Add WeChat powcoder

Problem 5 Regular Expressions.

- Define a regular expression for all strings of odd length, over the alphabet of $\{0\}$.
- Define a regular expression for identifiers over the alphabet of $\{A,B,C,a,b,c,0,1,2,3,4,5,6,7,8,9\}$, such that an identifier must begin with an alphabetic character and must contain at least one numeric character.

Problem 6 Context free grammar.

Consider the following context free grammar:

$$\begin{aligned} P ::= & id \mid P \vee P \mid P \wedge P \mid \neg P \mid (P) \\ id ::= & p \mid q \mid r \mid s \end{aligned}$$

where the set of terminals symbols is $\{p, q, r, s, \vee, \wedge, \neg, (,)\}$ and the start symbol is P .

- Give the leftmost derivations for the following expression:

- $\neg(p \wedge q)$

- Show that this language is ambiguous.

Problem 7 Context Free Grammar.

For each of the following grammars state whether it is ambiguous or unambiguous. If it is ambiguous give an equivalent unambiguous grammar. If it is unambiguous, state all the precedences and associativities enforced by the grammar. Assume Id generates identifiers.

a) $E ::= E + F \mid E - F \mid F$

$F ::= -F \mid Id \mid (E)$

b) $E ::= E \vee F \mid F$

$F ::= G \cap F \mid G \cup F \mid G$

$G ::= G \wedge H \mid H$

$H ::= Id \mid (E)$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

a) What are the free variables in the λ -term $(\lambda x . (\lambda y . y z)) y x$.

two different
y

① add parenthesis

$$(\lambda x . ((\lambda y . (y z)) y)) x.$$

② $FV(x) = \{x\}$

$$FV(\lambda x. t), FV(t) - \{x\}$$

$$FV(t_1, t_2) = FV(t_1) \cup FV(t_2)$$

$$FV((\lambda y . (y z)) y)$$

$$= FV(\lambda y . (y z)) \cup FV(y)$$

$$= (FV(y z) - \{y\}) \cup \{y\}$$

$$= (FV(z) - \{y\}) \cup \{y\} \cup \{z\}$$

$$= \{y, z\}$$

$$FV((\lambda x. A)x)$$

$$= FV(\lambda x. A) \cup FV(x)$$

$$= (FV(A) - \{x\}) \cup \{x\}$$

$$= \{x, y, z\}$$

Assignment Project Exam Help

<https://powcoder.com>

b) Fully evaluate $(\lambda x . ((\lambda y . (y z)) x)) z$.

= $\lambda x . ((\lambda y . (y z)) x)$ free variable

$$= \lambda y . ((\lambda y . y z) x)$$

$$= \lambda y . (y z)$$

$$= \lambda y . z$$

c) Define a function doubleChurch, which takes a Church Numeral and doubles it. Try to build it from scratch, without using the encoding of + or * as defined in lecture.

Input $\underline{\lambda}$
Output $\underline{\lambda f \lambda z. f(f \dots (f z))}$

n
 $2n$

$$\underline{\lambda n. f(n f z)}$$

$$\underline{\lambda n. \lambda f. \lambda z. (n f (n f z))}$$

Problem 5 [7pt] Consider the following grammar:

$G ::= S$
 $S ::= A M$
 $A ::= a E \mid b A A$
 $M ::= S \mid \epsilon$
 $E ::= a B \mid b A \mid \epsilon$
 $B ::= b E \mid a B B$

$G \rightarrow S \rightarrow A M \rightarrow A S \rightarrow A A M$

:

where a and b are terminals for letters a and b respectively. Define in English the language that the grammar generates.

$G \rightarrow S \rightarrow A M \rightarrow a E M \cdot a$

"a" is legal

$G \rightarrow S \rightarrow A M \rightarrow a b A \epsilon \rightarrow a b a \epsilon \rightarrow a b a$

"aba" is legal

$G \rightarrow S \rightarrow A M \rightarrow b A A \rightarrow a E a E \rightarrow b a a$

"baa" is legal

pattern: more a's than b's

Assignment Project Exam Help

$\epsilon: \#a = \#b$

<https://powcoder.com>

Add WeChat powcoder

#19

Assignment Project Exam Help
Names, Scopes, Bindings

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Static Scoping

- ① build symbol table for each scope
- ② build a tree: child is defined in the scope of parent
- ③ find name

(global)

Name	Kind	Type
n	Id	int
first	fun	void ->void
second	fun	Void->void

global

first

second

```
int n=2;  
void first() {  
    n = 1  
}  
  
void second() {  
    int n=0;  
    first();  
}  
  
first();  
second();
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Name	Kind	Type
n	Id	int

(first)

(second)

Name	Kind	Type
n	Id	int

Symbol tables determined at compile time
Search for the binding from the table
for the current scope

Dynamic Scoping

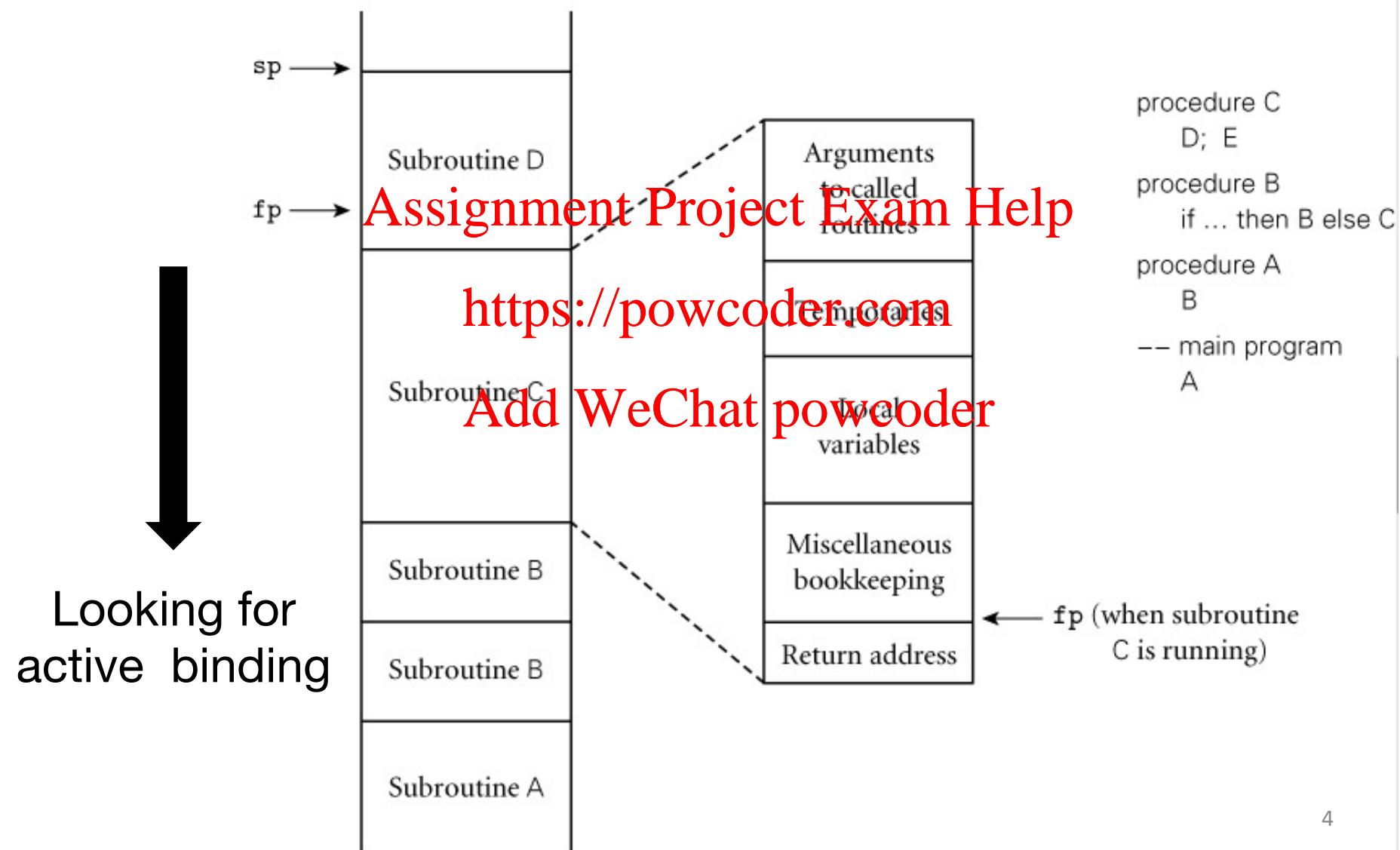
Static Scoping

- Bindings determined by lexical structure
[Assignment Project Exam Help](#)
- Local renaming principle
<https://powcoder.com>

Dynamic Scoping [Add WeChat powcoder](#)

- Bindings depend on flow of control
- Always use most recent, active binding
- Names important!
- Meaning of a variable can change

Dynamic Scoping and Stack



- ① build symbol table for each scope
 ② build tree: parent is the caller of the child
 ③ find name

Dynamic Scoping

(global)

Name	Kind	Type
n	Id	int
first	fun	void ->void
second	fun	Void->void

```
int n=2;
```

```
void first() {
    n = 1
}
```

```
void second() {
    int n=0;
    first();
}
```

```
first();
second();
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Set global n to be 1

Name	Kind	Type
------	------	------

Dynamic Scoping

(global)

```
int n=2;  
void first() {  
    n = 1  
}  
void second() {  
    int n=0;  
    first();  
}  
  
first();  
second();
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Name	Kind	Type
n	Id	int
first	fun	void ->void
second	fun	Void->void

Name	Kind	Type
n	Id	int

Name	Kind	Type
		(first)

Set n in “second” to be 1

Symbol tables changes at run time!
Always use most recent, active binding

Static vs Dynamic Scoping

Similarity

- One symbol table per scope
- Names are resolved from bottom to top in symbol table tree

Add WeChat powcoder

Difference

- Static: the symbol table tree is stable
- Dynamic: the symbol table tree changes during execution

If a subroutine is passed as a parameter, how do we resolve names?

```
function F(int x) {  
    function G(fx) {  
        int x = 13;  
        fx();  
    }  
    https://powcoder.com  
    function H() {  
        Add WeChat powcoder  
        print(x); // Which x?  
    }  
    G(H);  
}
```

What's the parent of H's symbol table?

Who's the parent of H's symbol table?

```
function F(int x) {  
    function G(fx) {  
        int x = 13;  
        fx();  
    }  
    function H() {  
        print(x);  
    }  
    G(H);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Shallow Binding: when
the subroutine is called

Deep Binding: when
reference is created

Shallow Binding: use the environment at call time

(F)

```
function F(int x) {  
    function G(f) {  
        int x = 13;  
        fx();  
    } ;  
    function H() {  
        print x;  
    } ;  
    G(H);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Name	Kind	Type
x	para	int
G	fun	fun ->void
H	fun	void->void

(G)

Name	Kind	Type
fx	para	fun
x	Id	int

Established
at call time

(H)

Name	Kind	Type

Deep Binding: use the environment when ref. is created

Assignment Project Exam Help
<https://powcoder.com>

```
function F(int x) {  
    function G(f) {  
        int x = 13;  
        fx();  Add WeChat powcoder  
    } ;  
    function H() {  
        print x;  
    } ;  
    G(H);  
};
```

Established when
ref. is created

Name	Kind	Type
x	para	int
G	fun	fun ->void
H	fun	void->void

Name	Kind	Type
------	------	------

Impl. of Deep Binding: Function Closures

Assignment Project Exam Help

```
function F(int x) {  
    function G(f) {  
        int x = 13;  
        fx(); Add WeChat powcoder  
    };  
    function H() {  
        print x;  
    };  
    G(H);  
};
```

- A function closure contains:
- A pointer to the function
 - The current symbol table (or all symbol tables, depending on implementation)

Pass in a function closure

Shallow, Deep Binding and Static, Dynamic Scoping

Dynamic scoping

- Both shallow and deep binding are implemented
[Assignment Project Exam Help](#)
- Deep binding has a higher cost at run time
<https://powcoder.com>

Static scoping

[Add WeChat powcoder](#)

- Shallow binding has never been implemented

#20

Assignment Project Exam Help
Names, Scopes, Bindings

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

```

void f (int i) {
    int a;
    void h (int j) {
        a = j;
    }
    void g (int k) {
        h(k);
    }
    g(i+2);
}

```

fun f

a	id
h	fun
g	fun
i	para

fun h

H	j	para
---	---	------

fun g

G	k	para
---	---	------

Assignment Project Exam Help

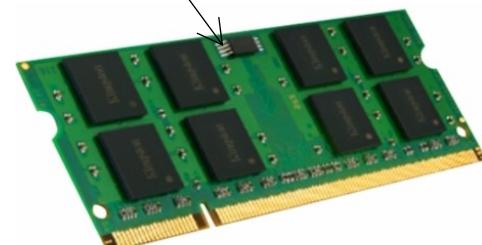
<https://powcoder.com>

Add WeChat powcoder

```

void f (int i) {
    int a;
    void h (int j) {
        a = j;
    }
    void g (int k) {
        h(k);
    }
    g(i+2);
}

```



memory

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Name	Kind
a	id
i	para
h	fun
g	fun

Name	Kind
k	para

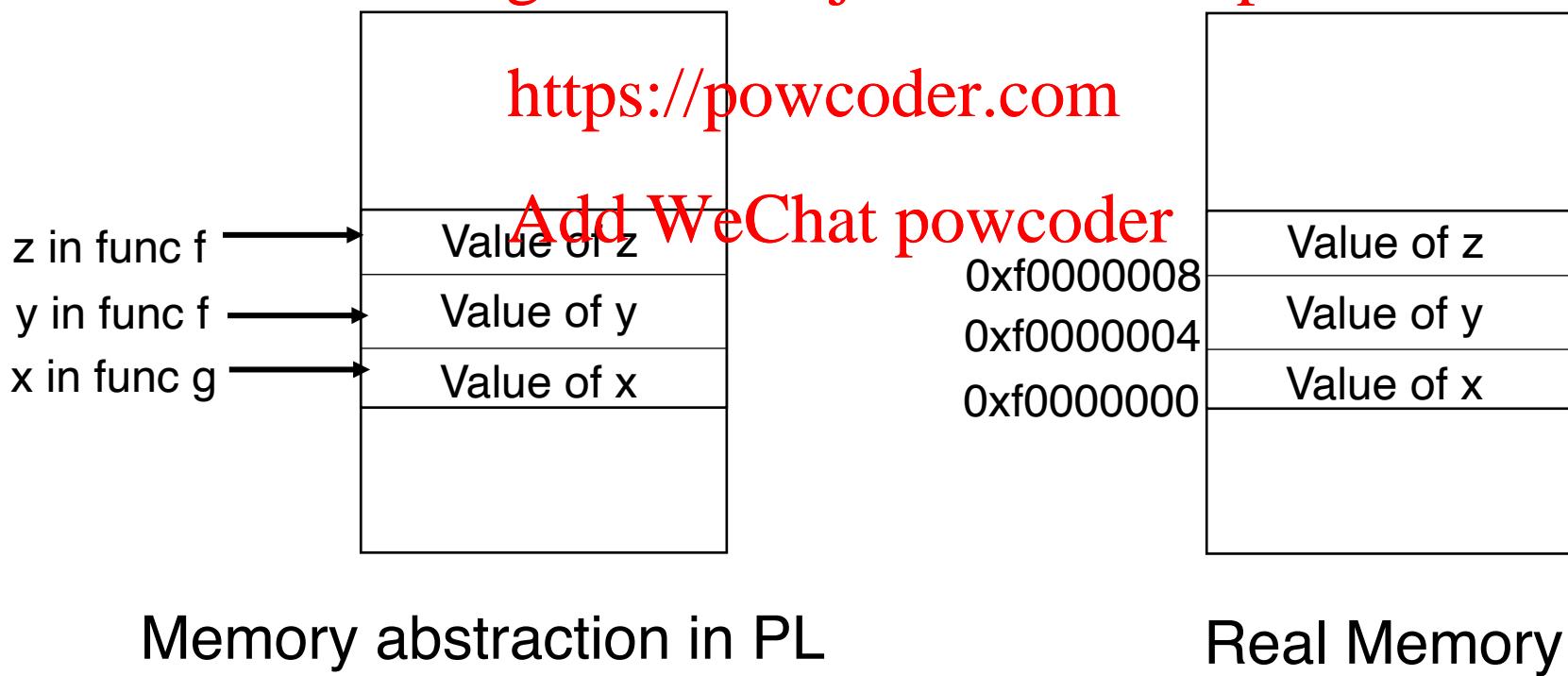
Name	Kind
j	para

Where is a? From symbol table of h, go up **two hops**

Establishing Addressability

How does the compiled code locate a variable used in a function at run time?

Assignment Project Exam Help



Establishing Addressability

How does the compiled code locate a variable used in a function at run time?

Assignment Project Exam Help

- What is the start address?
 - Start address = base address (of a scope) + offset (in the scope)

Add WeChat powcoder

Global variables

Local variables (w/ or w/o surrounding scopes)

Heap variables

Typical Code and Data Layout

When executed, a program occupies memory

Code section

- Stores source code
- Read-only

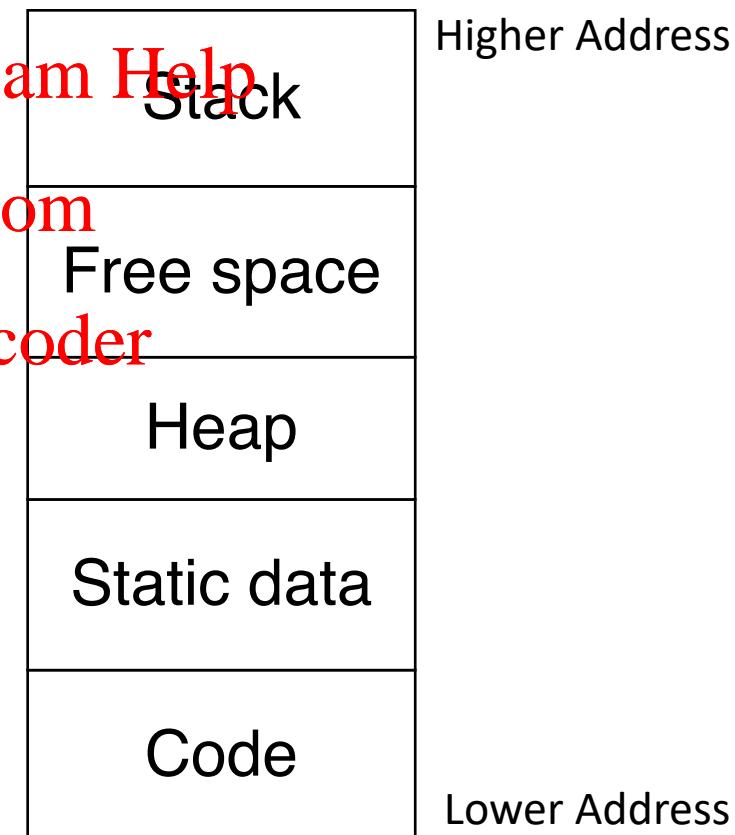
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Data section

- Static data area – *global*
- Stack
- Heap



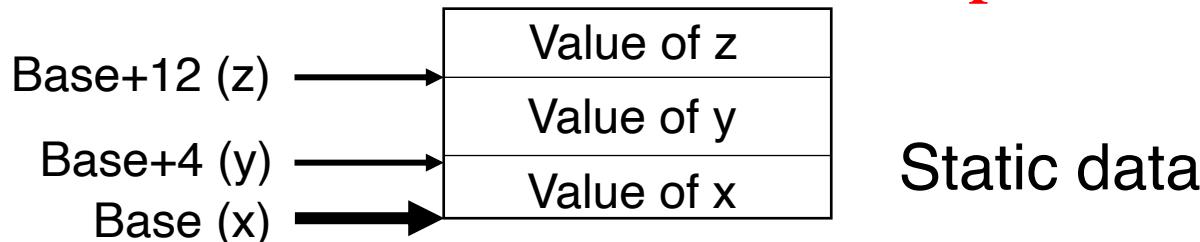
Global & Static Variables

Base Address: fixed throughout the run

Offset: known at compile time (length of each variable is determined by its type)

<https://powcoder.com>

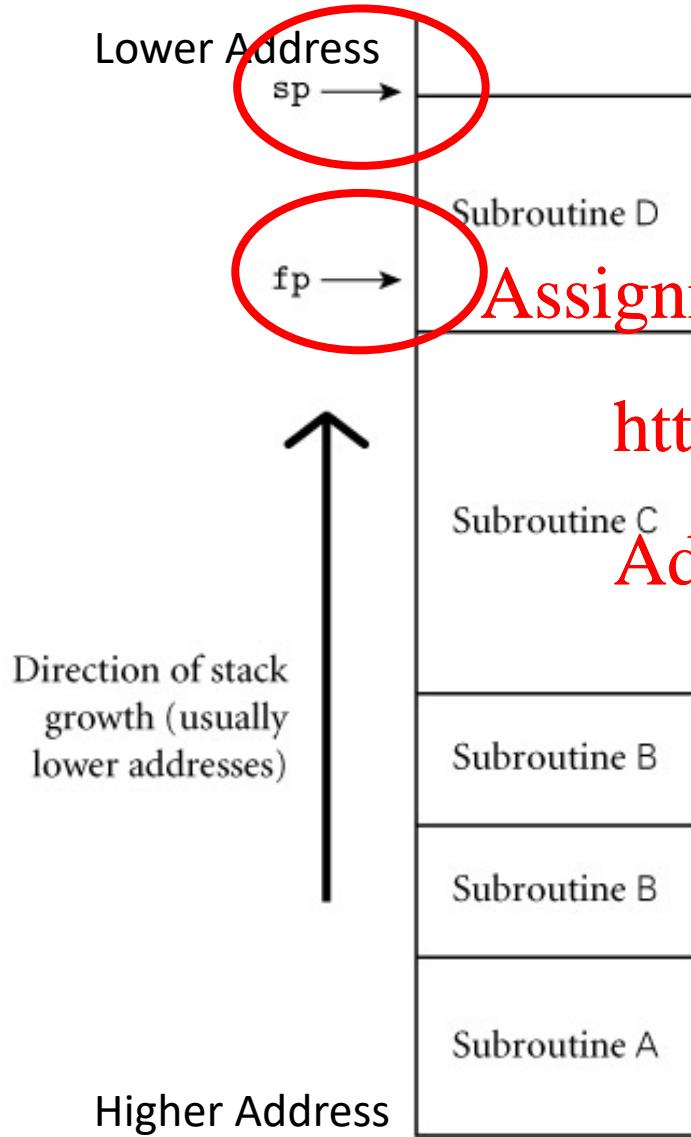
Add WeChat powcoder



Global variables:

```
int x;  
float y;  
int z;
```

Scope and Stack



procedure C

D; E

procedure B

if ... then B else C

procedure A

B

-- main program

<https://powcoder.com>

Each instance of a subroutine has
Add WeChat **powcoder**

- Compiler generates code that setup frame, call routine, and destroy frame

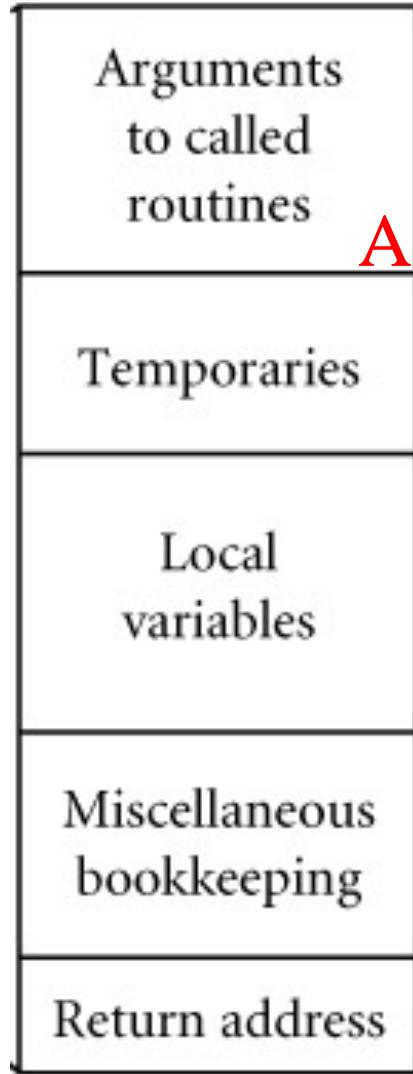
Frame pointer (fp): currently active frame
Stack pointer (sp): free space on stack

Hardware Support

X86 Registers and instructions:

- Stack pointer register: esp
[Assignment Project Exam Help](#)
- Frame pointer register: ebp
<https://powcoder.com>
- Push instructions: push, pusha, etc.
- Pop instructions: pop, popa, etc.
[Add WeChat powcoder](#)
- Call instruction: call
- Return instruction: ret

Typical Frame Layout



Assignment Project Exam Help

Temporaries.: register spill area, language-specific exception-handling context, and so on (not covered)
<https://powcoder.com>

Add WeChat powcoder

Bookkeeping info.: a reference to the stack frame of the caller (also called the **dynamic link**) and so on

← fp
(when the frame is active)

Local Variables

addr of x addr of y
fp + 4 - 4 fp + 4 + 0
base base

Arguments to called routines
Temporaries
Local variables
Miscellaneous bookkeeping
Return address

The offsets of objects *within* a frame usually can be statically determined

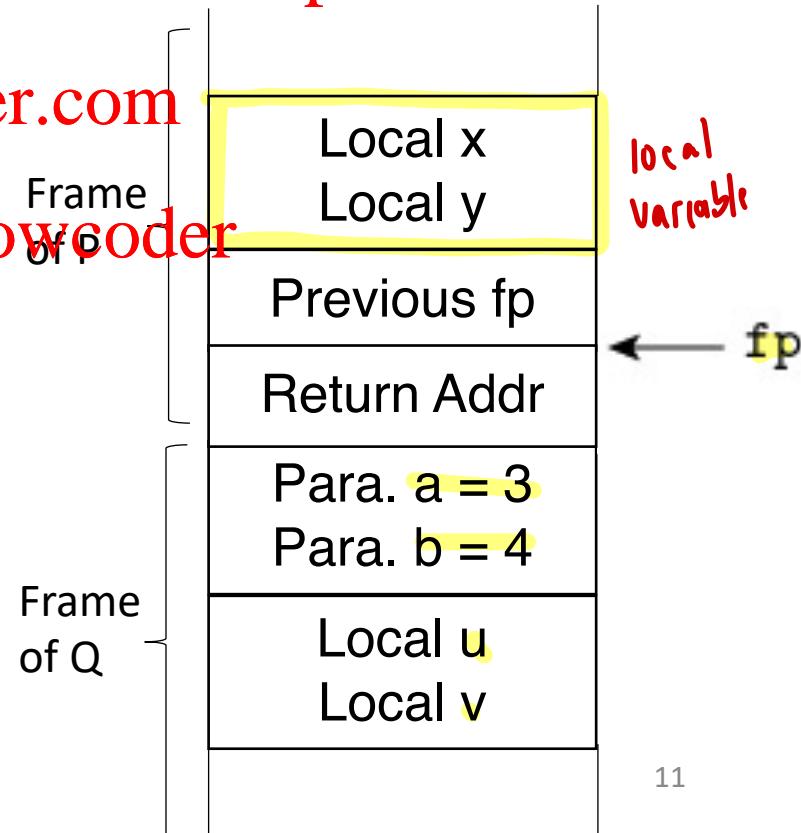
Assignment Project Exam Help

```
void P(int a, int b) {  
    int x, y;  
}  
void Q() {  
    int u, v;  
    P(3, 4);  
}
```

Add WeChat powcoder

```
void Q() {  
    int u, v;  
    P(3, 4);  
}
```

(when the frame is active)



Nested Scopes

Languages with nested functions:

How do we access local variables from other frames?

Assignment Project Exam Help

Static Scoping: <https://powcoder.com>

Static link: a pointer to the frame of enclosing function

Dynamic Scoping:

Dynamic link: pointer to the previous frame in the current execution (the same as previous fp)

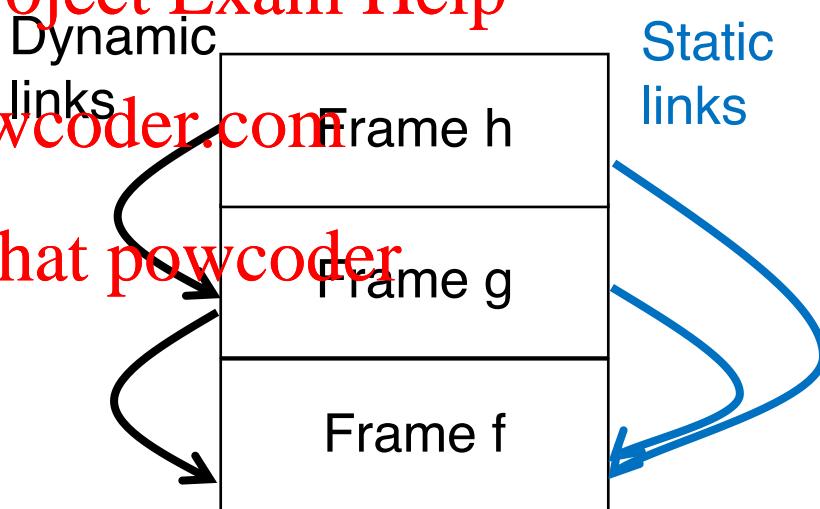
Static Link and Dynamic Link

```
void f (int i){  
    int a;  
    void h (int j){  
        a = j;  
    }  
    void g (int k){  
        h(k);  
    }  
    g(i+2);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

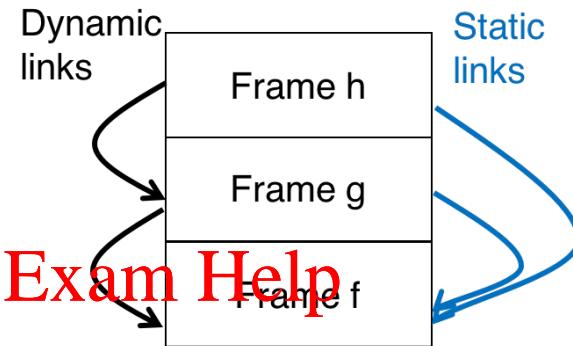
Add WeChat powcoder



Where is variable a in h?

Static Scoping

```
void f (int i) {  
    int a;  
    void h (int j) {  
        a = j;  
    }  
    void g (int k) {  
        h(k);  
        g(i+2);  
    }  
}
```



Assignment Project Exam Help

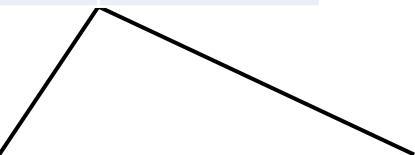
<https://powcoder.com>

Symbo tables

Add WeChat powcoder

Where is a? From frame of h, go up one hop following static links

Name	Kind
a	id
i	para
h	fun
g	fun



Name	Kind
k	para
j	para

Where is a? From symbol table of h, go up one hop

Where is variable a in h?

Dynamic Scoping

Name	Kind
a	id
i	para
h	fun
g	fun

Name	Kind
k	para

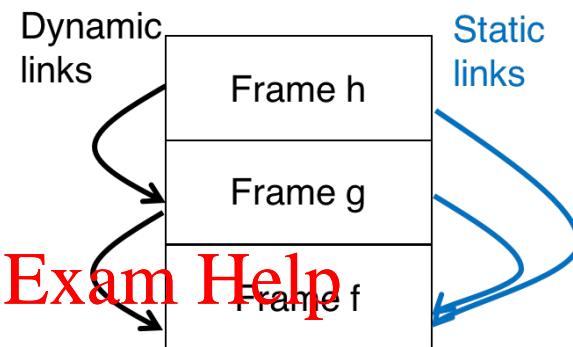
Name	Kind
j	para

```
void f (int i) {  
    int a;  
    void h (int j) {  
        a = j;  
    }  
    void g (int k) {  
        h(i+1);  
        g(i+2);  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Symbol tables
Add WeChat powcoder
Where is a? From frame of h,
go up two hops following dynamic links

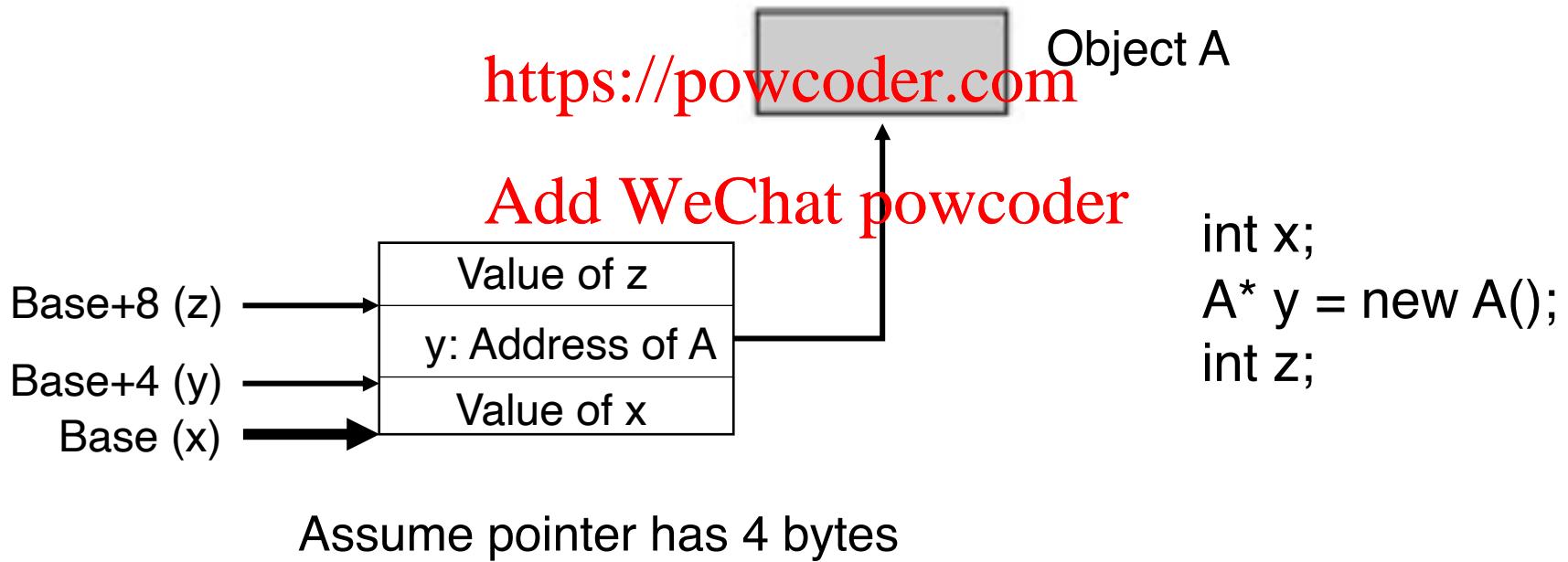


Where is a? From symbol table of h, go up two hops

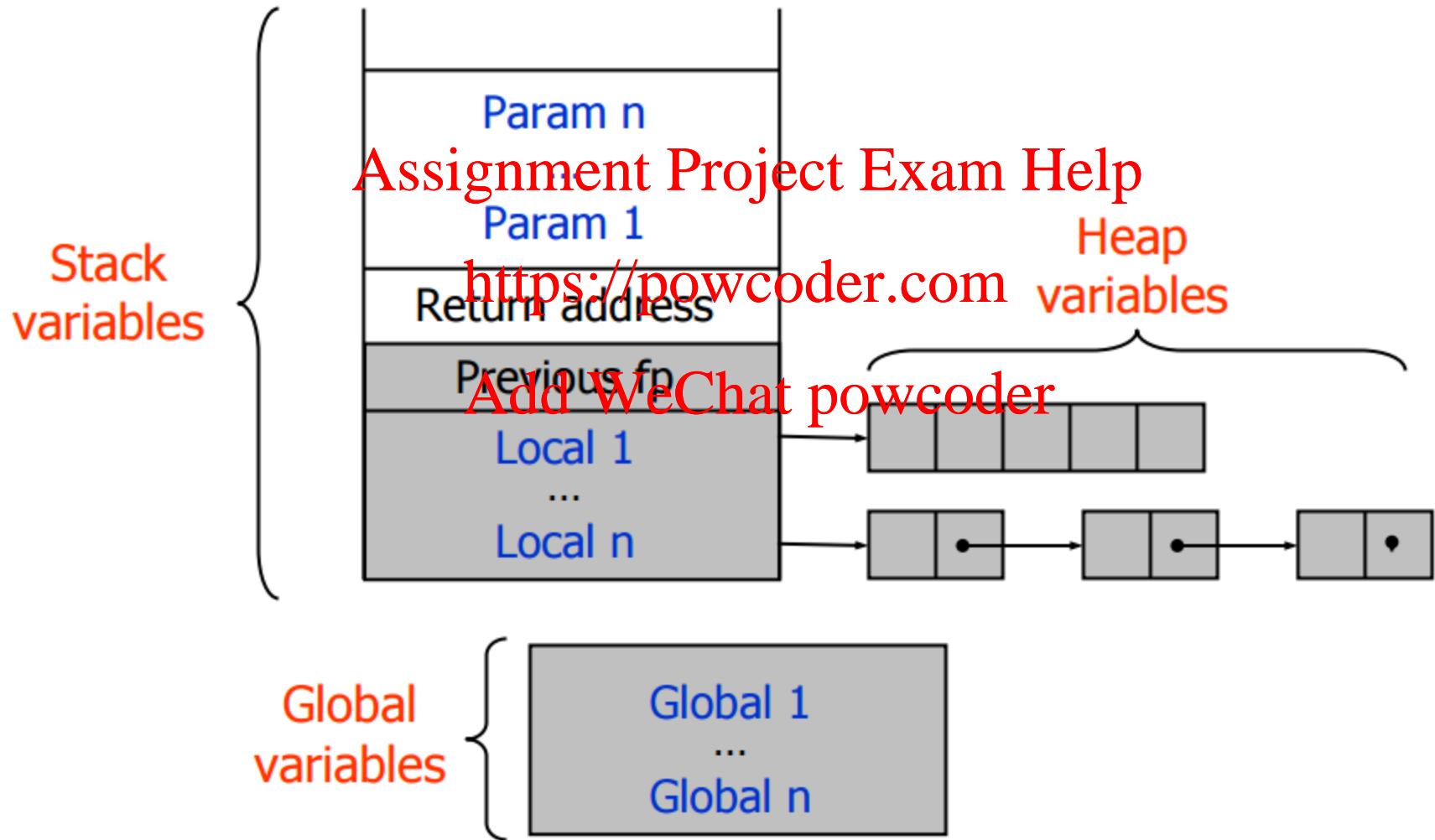
Heap-Based Variable

Similar to a normal variable, except that the memory cell stores the address of heap space

Assignment Project Exam Help



Big Picture: Data Memory Layout



#21

Assignment Project Exam Help
Procedures and Functions

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Terminology

Function (Fortran, Ada): returns a value

Assignment Project Exam Help

Procedure (Ada), *Subroutine* (Fortran): returns no value
<https://powcoder.com>

C-like language: both are called *function*
Add WeChat powcoder

Method (C++, Java): a function declared inside a class

Caller & Callee

When function A calls function B

- A is the *Caller*
- B is the *Callee*

Assignment Project Exam Help

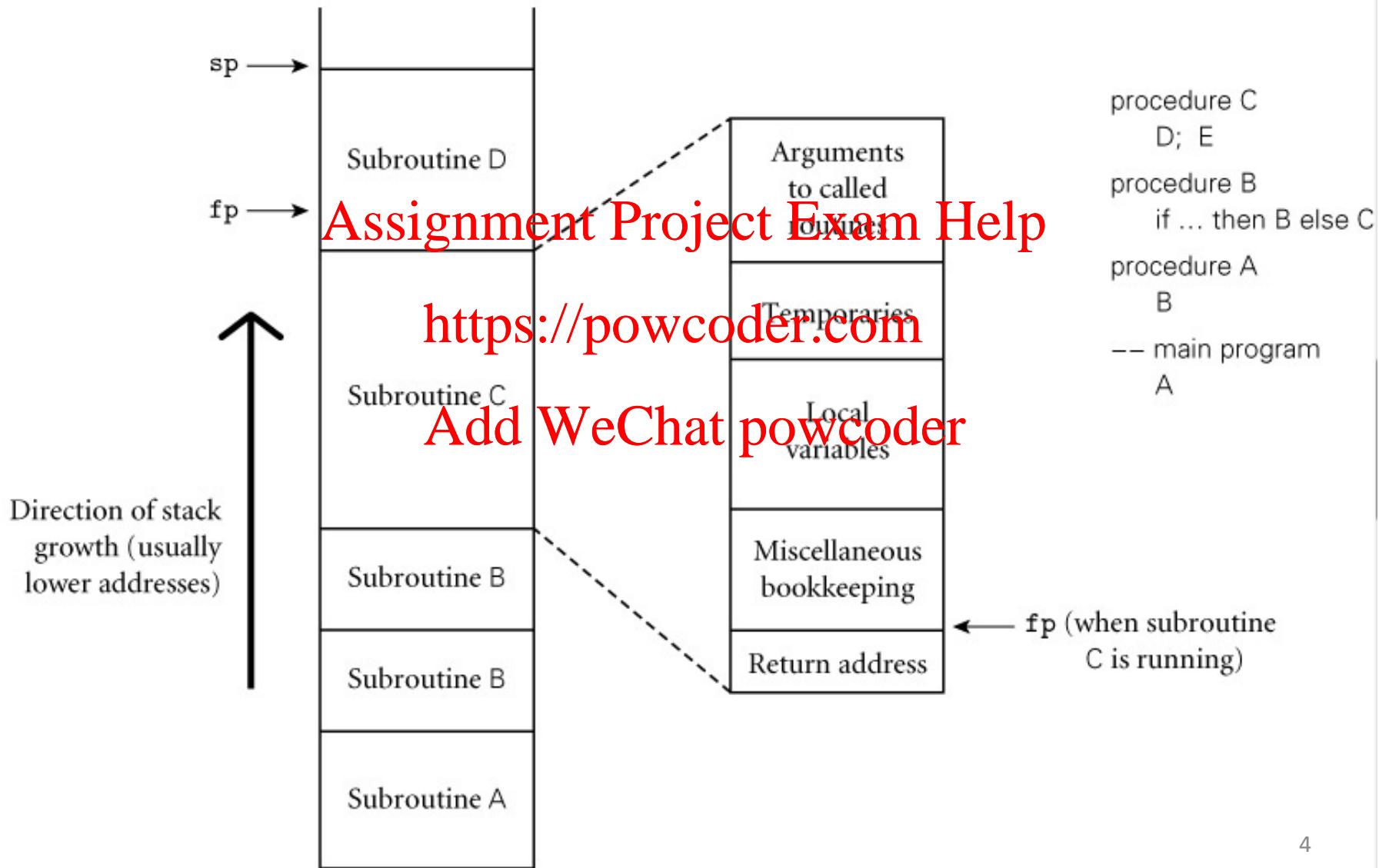
<https://powcoder.com>

What needs to be done?

Add WeChat powcoder

Who is responsible?

Stack-Based Allocation



Stack Frame (Activation Record)

A stack frame contains:

- Local variables
- Temporaries
- Arguments, return values
- Bookkeeping info

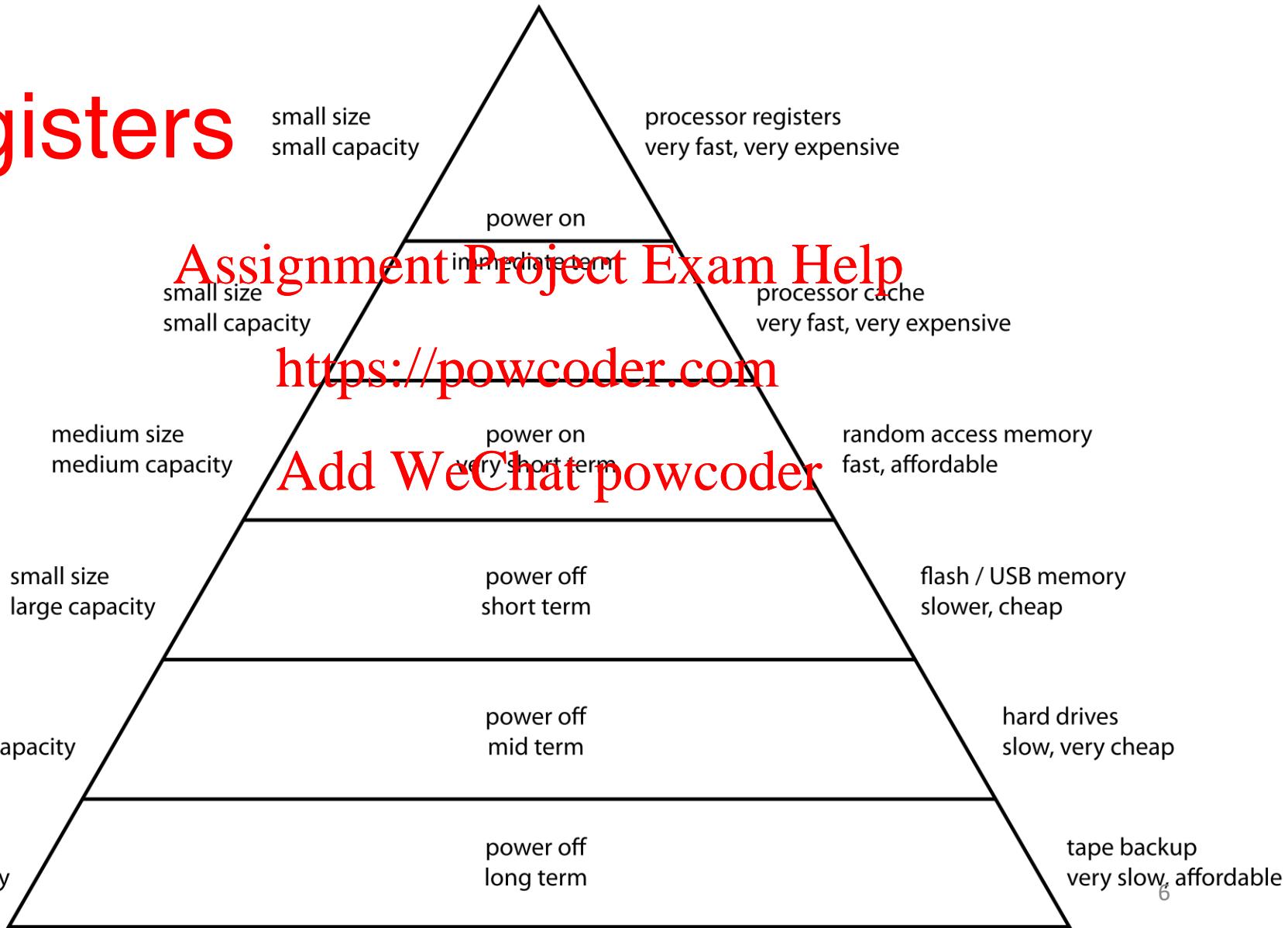
arguments
temporaries
locals
misc bookkeeping
return addr

When is it created? By whom?

When is it destroyed? By whom?

Computer Memory Hierarchy

Registers



X86 Registers

General registers

eax, ebx, ecx, edx

Assignment Project Exam Help

<https://powcoder.com>

Indexes and pointers

sp, pc, fp, ... Add WeChat powcoder

And many more ...

Registers are shared among function calls

Registers

How can registers be used?

Assignment Project Exam Help

When should they be used?

<https://powcoder.com>

Who saves registers?

Add WeChat powcoder

Saving & Restoring Registers

Registers are shared, caller & callee need to save and restore registers

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Ideally: save exactly <https://powcoder.com> registers used in caller & callee

[Add WeChat](#) [powcoder](#)

In practice, registers fall into two categories:

1. Caller-saves
2. Callee-saves

Caller Save vs. Callee Save

Caller saves all caller-saves registers before call

Callee saves all callee-saves registers before run

Assignment Project Exam Help

Callee restores all callee-saves registers before exit

Caller restores all caller-saves registers before resume

Add WeChat powcoder

Separation of responsibility:

1. Caller assumes callee will not destroy callee-saves
2. Callee assumes nothing is valuable in caller-saves

C Calling Convention on x86

Caller-Saves: EAX, ECX, EDX

Assignment Project Exam Help

Callee-Saves: EBX, EDI (dest. idx), ESI (src. idx)
<https://powcoder.com>

Return value stored in EAX
Add WeChat powcoder

What needs to be done before P calls Q?

- Saving dynamic link (current fp)
- Saving caller-saves registers
- Changing stack and frame pointers

Add WeChat powcoder

What needs to be done before Q returns to P?

- Restoring saved registers
- Deallocating stack space of Q

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Control Flows in HW

Program Counter (PC): a register that always contains the memory address of an instruction

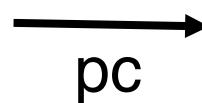
[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

PC always points to the instruction following the one being executed

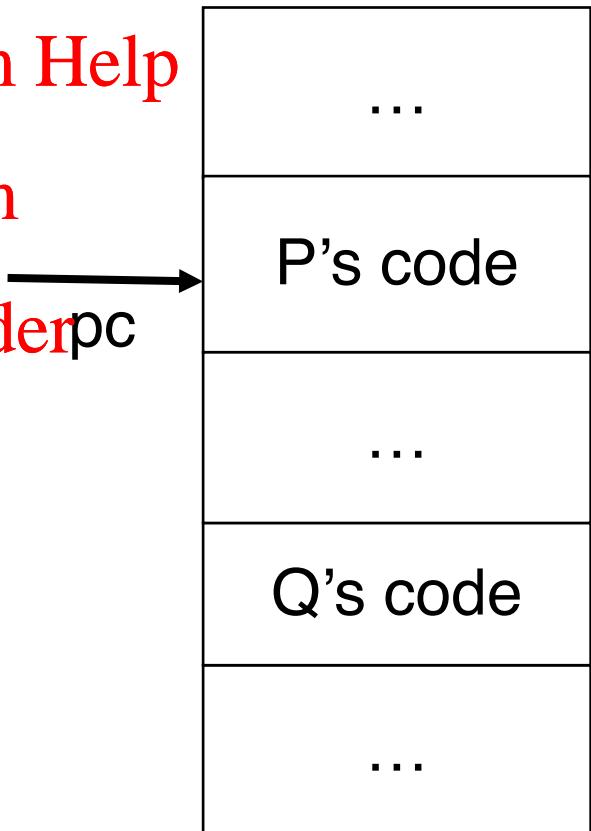
[Add WeChat](#) [powcoder](#)

PC can be modified by *jump*, *call*, *return* instructions



What needs to be done before P calls Q?

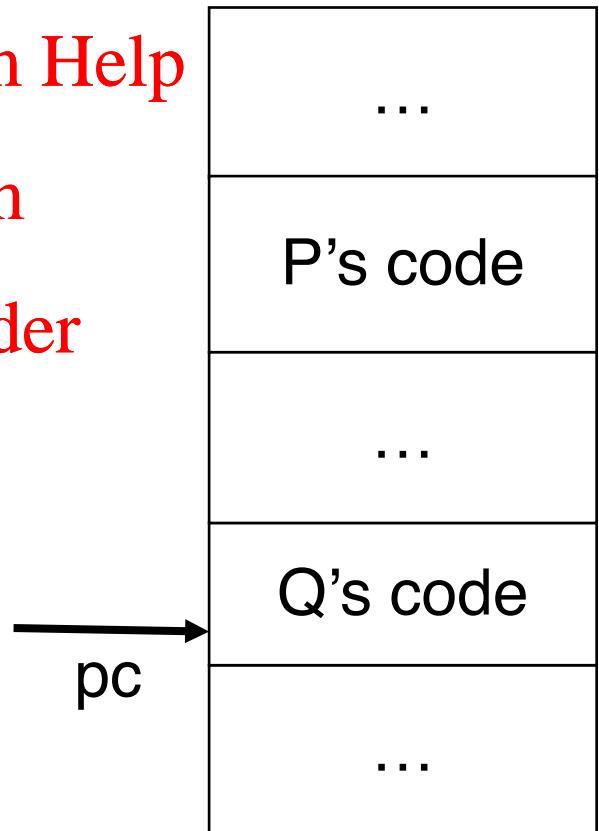
- Saving dynamic link (current fp)
- Saving other registers
- Changing stack and frame pointer
- Saving return address
- Changing program counter



What needs to be done before Q returns to P?

- Restoring saved registers
- Deallocating stack space of Q
- **Restoring program counter**

<https://powcoder.com>
Add WeChat powcoder



Parameter Passing

Caller passes parameters to callee

Callee passes values to caller

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

Formal vs. actual parameters

Add WeChat powcoder

Parameter passing modes

Value or reference?

Input or output? [Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

Read or write?

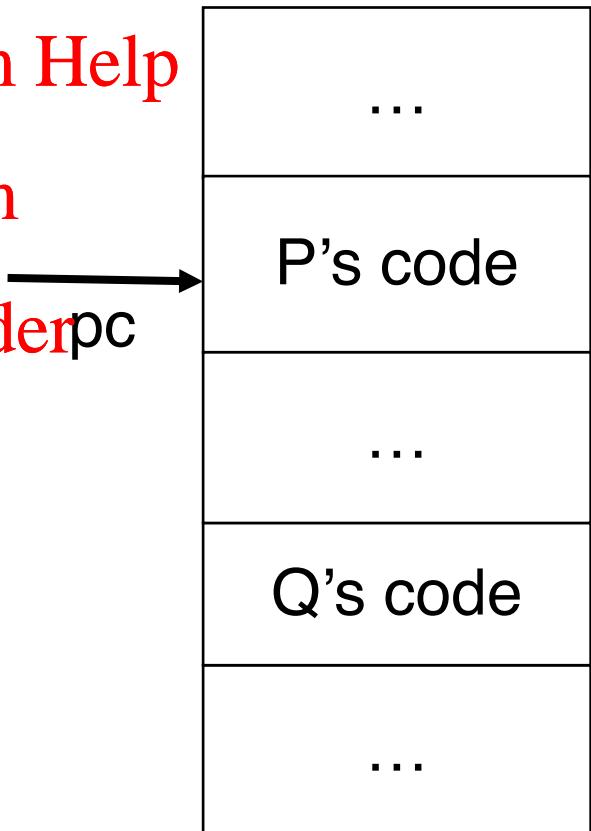
[Add WeChat powcoder](#)

How is parameter passing implemented?

(covered later)

What needs to be done before P calls Q?

- Saving dynamic link (current fp)
- Saving other registers
- Changing stack and frame pointer
- Saving return address
- Changing program counter
- **Passing parameters**



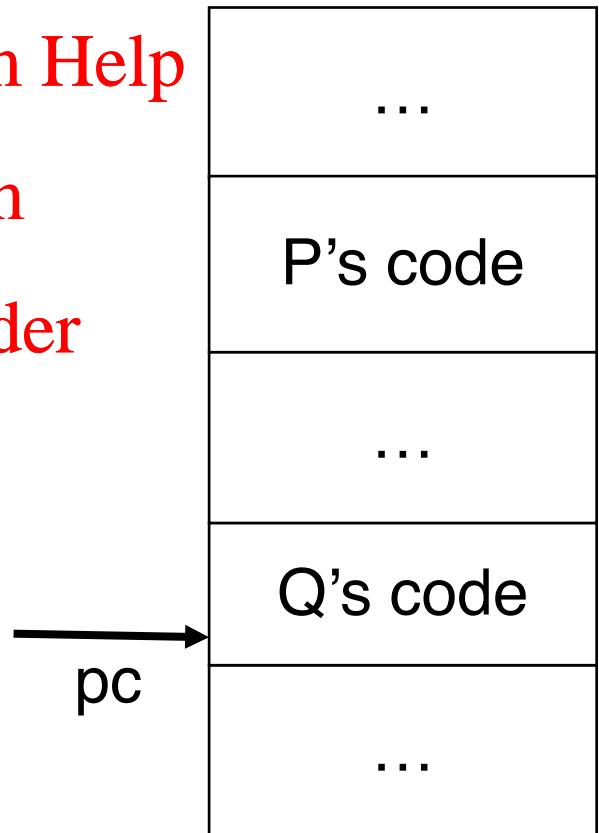
What needs to be done before Q returns to P?

- Restoring saved registers
- Deallocating stack space of Q
- Restoring program counter
- **Passing return values**

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



One Example (implementation dependent)

- Caller processes actual parameters (evaluation, address calculation) and stores them
- Caller stores some control information (e.g., return address) and registers
- Control transferred from Caller to Callee
- Callee allocates storage for locals
- Callee executes Add WeChat powcoder
- Callee deallocates storage for locals
- Callee stores return value (if function)
- Control transferred back to Caller
- Caller deallocates storage used for control information and actual parameters
- Caller restores registers

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

A Typical Calling Sequence

Caller

saves registers
Assignment Project Exam Help
computes values of parameters, move them to stack or registers
switch control to callee (call)
<https://powcoder.com>

Callee Prologue Add WeChat powcoder

allocates frame (change sp)
saves previous frame pointer (dynamic link) and sets new one
saves registers might be overwritten in the callee

A Typical Calling Sequence

Callee Epilogue

moves return value into stack or register
Assignment Project Exam Help
restore callee-saves registers
restore fp and sp <https://powcoder.com>
transfer control to caller

Add WeChat powcoder

Caller

moves return value where needed
restores caller-saves registers when needed

Function call is space and time consuming

Space: each active function requires one frame
(activation record) on stack

<https://powcoder.com>

Time: instructions are added by the compiler to set up and clean up function calls (calling sequence)

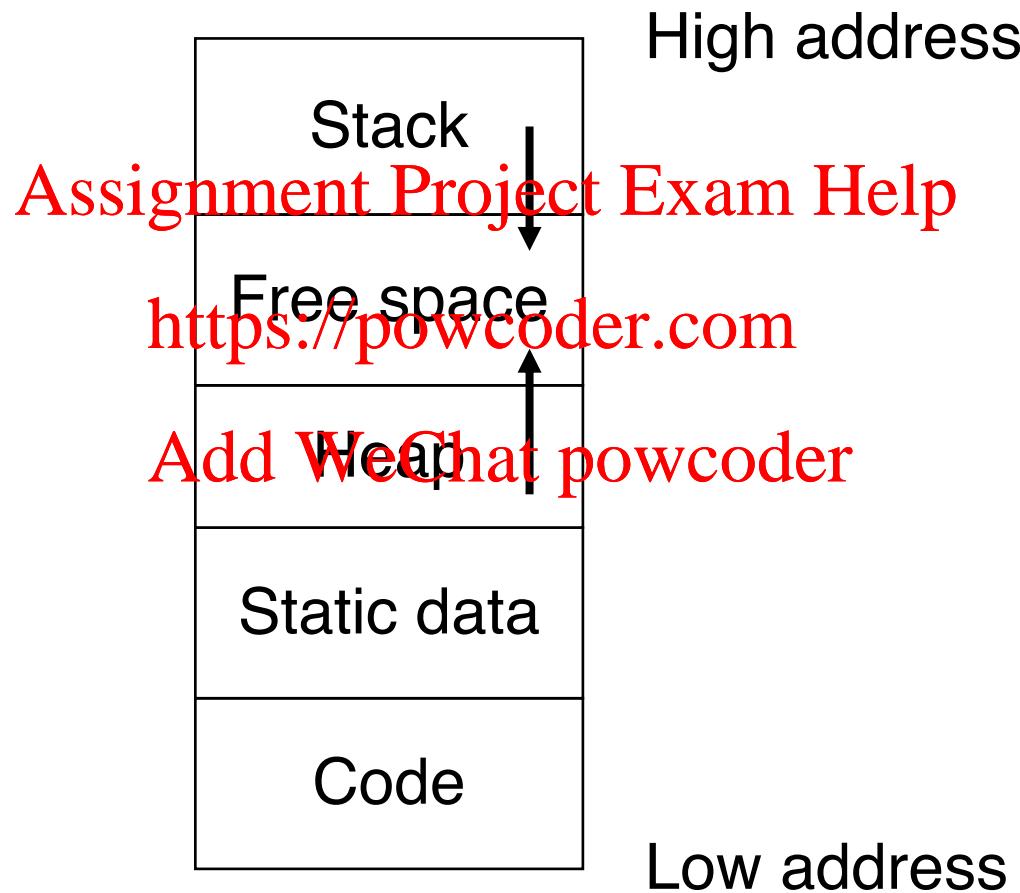
#22

Assignment Project Exam Help
Procedures and Functions
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Review of Storage Layout



Support for Functions

```
int fact(int n) {  
    int tmp = 0;  
    if n==1 return 1;  
    else {tmp = n-1;  
          return n*fact(tmp);}  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



During execution, ***each function call has one frame*** (activation record) on the stack

Stack Frame (Activation Record)

A stack frame contains:

- Local variables
- Temporaries
- Return values
- Bookkeeping info
- Arguments (to the callee)

arguments
temporaries
locals
misc bookkeeping
return addr

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Stack Frame (Activation Record)

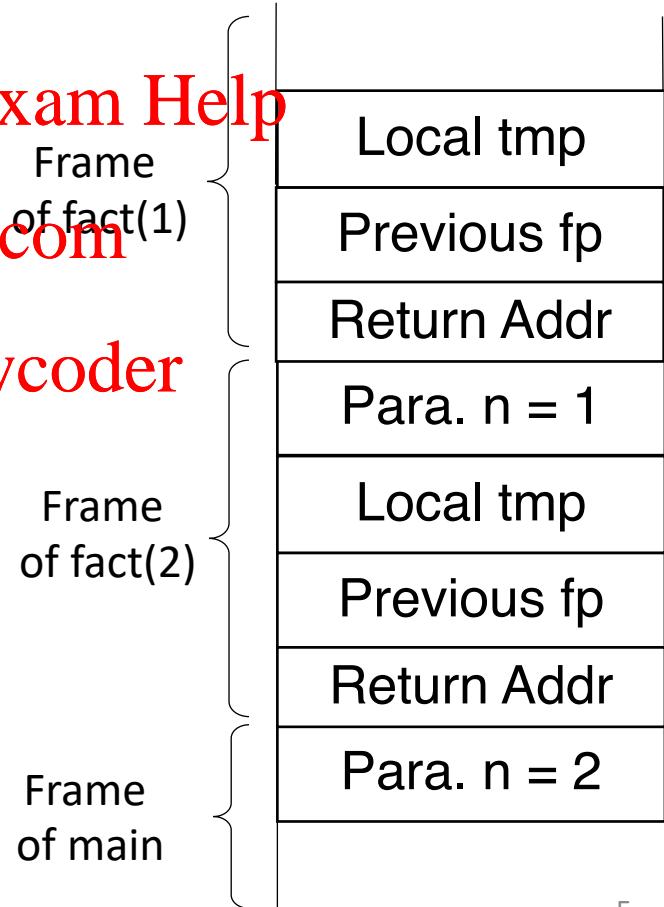
```
int fact(int n) {  
    int tmp = 0;  
    if n==1 return 1;  
    else {tmp = n-1;  
          return n*fact(tmp);}  
}
```

Calling fact(2) in main

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



```
int fact(int n) {  
    int tmp = 0;  
    if n==1 return 1;  
    else {tmp = n-1;  
          return n*fact(tmp);}  
}
```

What happens for fact (100) ?

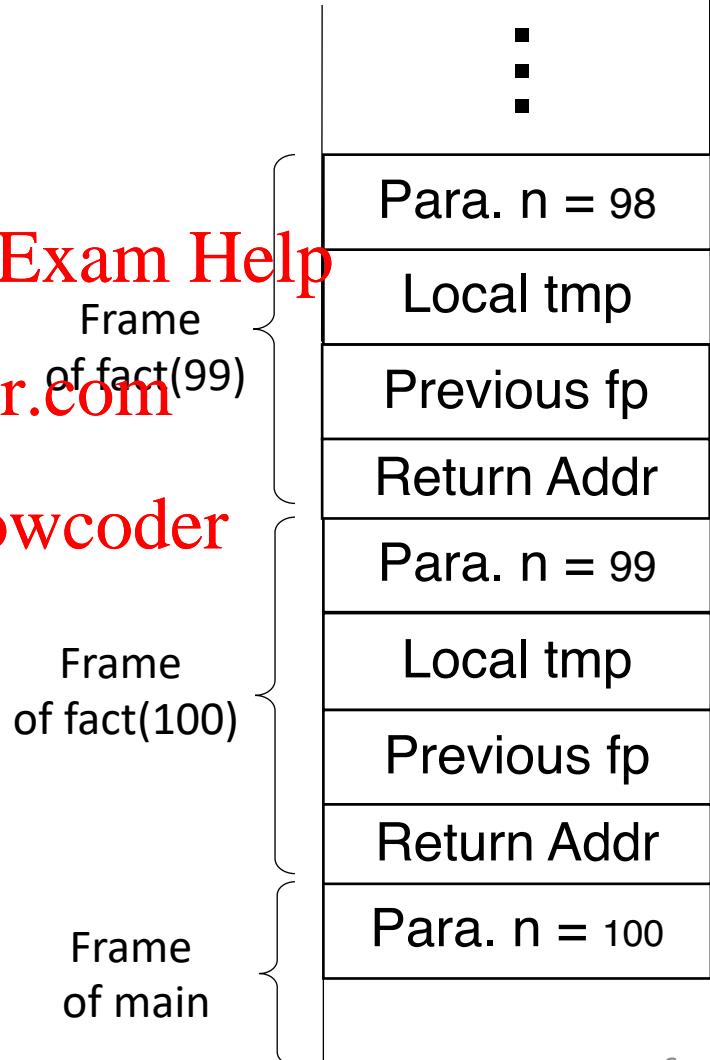
Consumes memory

Wastes time on calling sequence

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



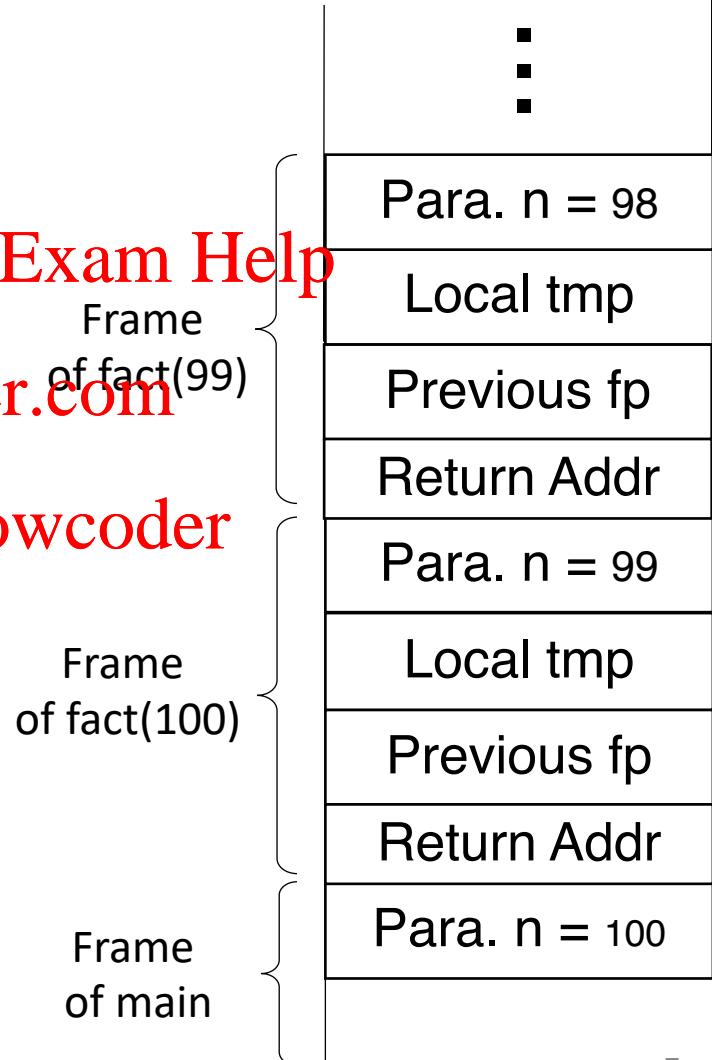
```
int fact(int n) {  
    int tmp = 0;  
    if n==1 return 1;  
    else {tmp = n-1;  
          return n*fact(tmp);}  
}
```

Can we destroy the frame of
fact(100) before going into
fact(99)?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



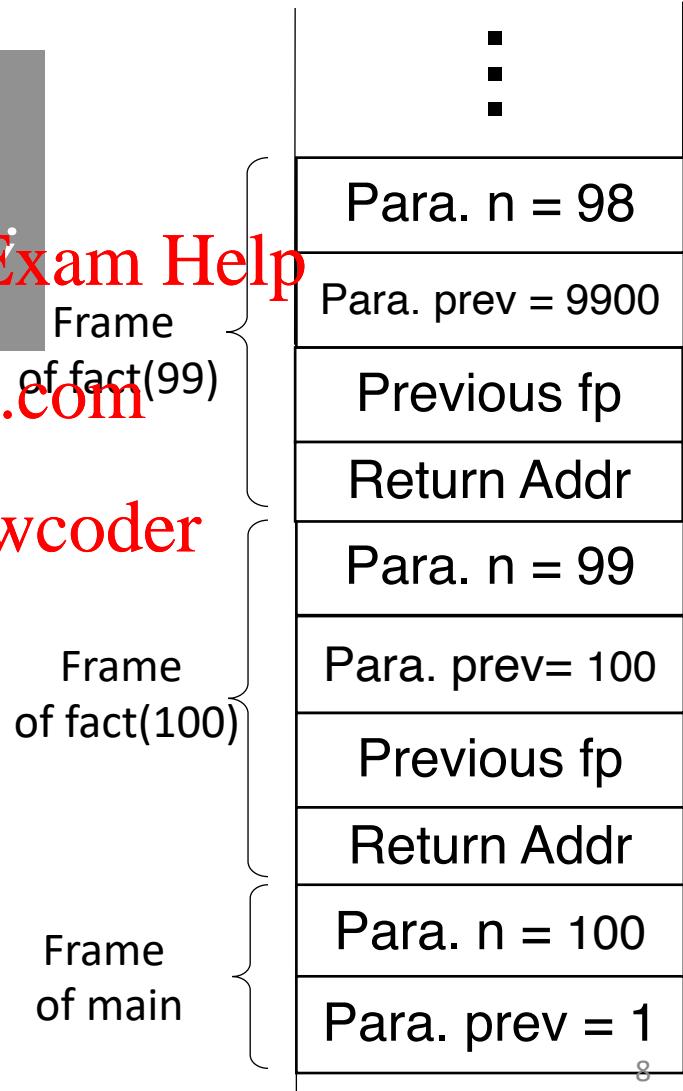
A Different Implementation

```
int fact(int n, int prev ) {  
    if ( n == 1 ) return prev ;  
    else return fact(n-1, prev*n);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Can we destroy the frame of
fact(100) before going into
fact(99)?



A Different Implementation

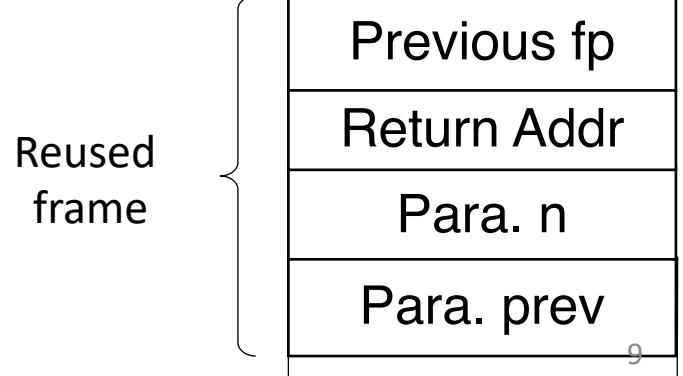
```
int fact(int n, int prev ) {  
    if ( n == 1 ) return prev ;  
    else return fact(n-1, prev*n);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Can we destroy the frame of
Add WeChat powcoder
fact(100) before going into
fact(99)?

What change enables the
more efficient implementation?



Tail-Recursive Functions

Recursion only occur at the end of a function: no computation after recursive call

Assignment Project Exam Help

```
int fact(int n, int prev){  
    if ( n == 1 ) return prev ;  
    else return fact(n-1, prev*n) ;  
}
```

Add WeChat powcoder

The current activation record before recursive call is useless!

Tail-Recursive Functions

Tail-recursive functions are equivalent to loops

```
int fact(int n, int prev) {  
    if ( n == 1 ) return prev ;  
    else return fact(n-1, prev*n) ;  
}
```

Add WeChat powcoder



```
int fact(int n, int prev) {  
    while (true) {  
        if ( n == 1 ) return prev ;  
        else {prev = prev*n; n--;}  
    } }
```

Is this function tail recursive?

```
int search(int[] a, int k, int fst, int lst) {  
    if ( fst == lst ) return (a[fst]==k?fst:-1);  
    int m = (fst + lst)/2;  
    if (k <= a[m]) return search(a, k, fst, m);  
    else return search(a, k, m+1, lst);  
}
```

Loop version

```
int binSearch(int[] a, int k, int fst, int lst) {  
    while (true) {  
        if (fst == lst) return (a[fst] == k ? fst : -1);  
        int m = (fst + lst) / 2;  
        if (k <= a[m], lst = m; Add WeChat;powcoder  
        else fst = m+1;  
    }  
}
```

Tail-Recursive Functions to Loops

Tail-recursive

```
int fact(int n, int prev ) {  
    if ( n == 1 ) return prev ;  
    else return fact(n-1,prev*n);  
}
```

unoptimized



Loops

```
int fact(int n, int prev ) {  
    while (true) {  
        if (n == 1) return prev ;  
        else {prev = prev*n; n--;}  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>



Add WeChat powcoder

In C and Java, the loop version is more efficient, though some compilers will automatically generate more efficient code

In most functional languages (e.g., Scheme, Ocaml), the compiler automatically generate code as efficient as the loop version

#23

Assignment Project Exam Help
Procedures and Functions
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Parameter Passing

How caller communicates with callee

Assignment Project Exam Help

Formal parameters: names in the declaration of a function
<https://powcoder.com>

Actual parameters: variables / expressions passed to a function

formal

```
foo (int x) { ... }  
foo (3+5);
```

actual

Parameter Modes

Call-By-Value (CBV)

Call-By-Value-Return (CBVR)
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Call-By-Reference (CBR)

Call-By-Name (CBN)

<https://powcoder.com>
Add WeChat powcoder

Call-By-Value

Calling mechanism

- Arguments are evaluated to their values
- Memory or registers allocated for arguments on AR
- Argument **values** copied to AR
- AR destroyed when callee returns

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

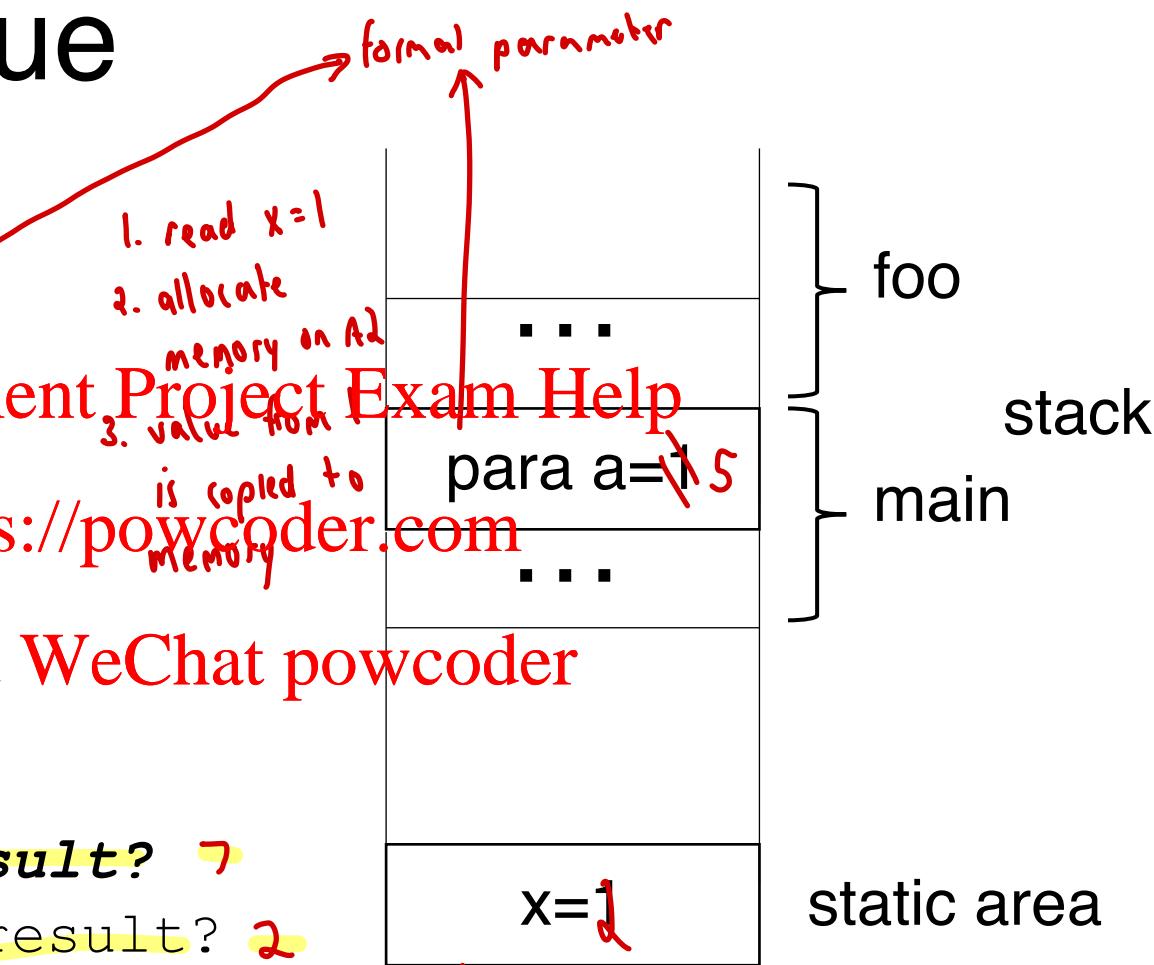
actual and formal parameter have separate memory.
Their values might be different

Call-By-Value

```
int x=1;
int foo (int a) {
    → x = 2;
    → print (a);
    → a = 5;           update value 5
    return x+a;
}
void main() {
    foo(x); //result? 7
    print(x); //result? 2
}
```

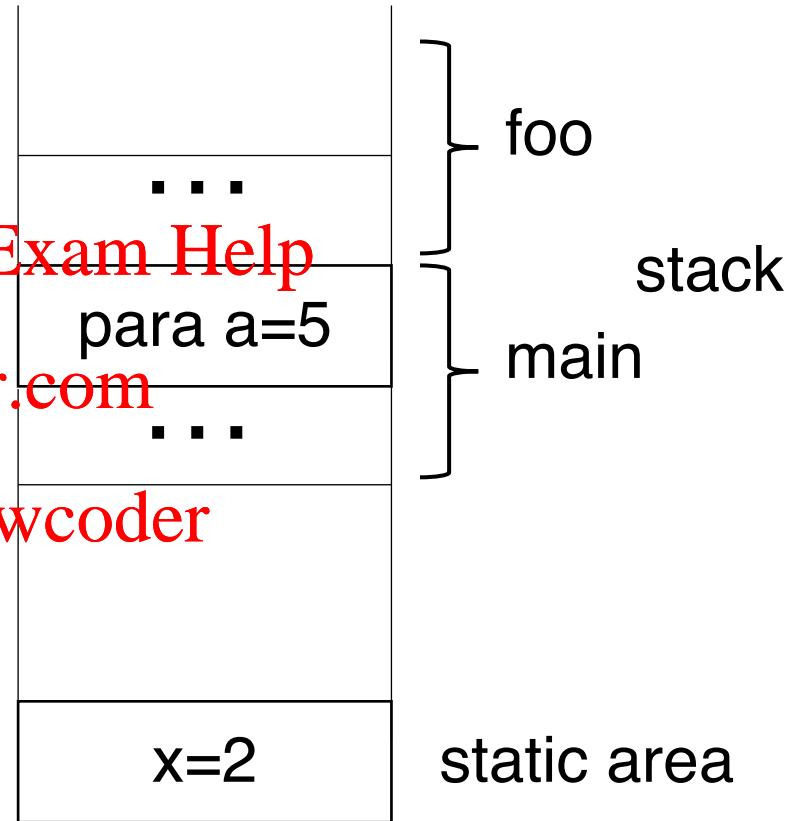
only updates

formal parameters



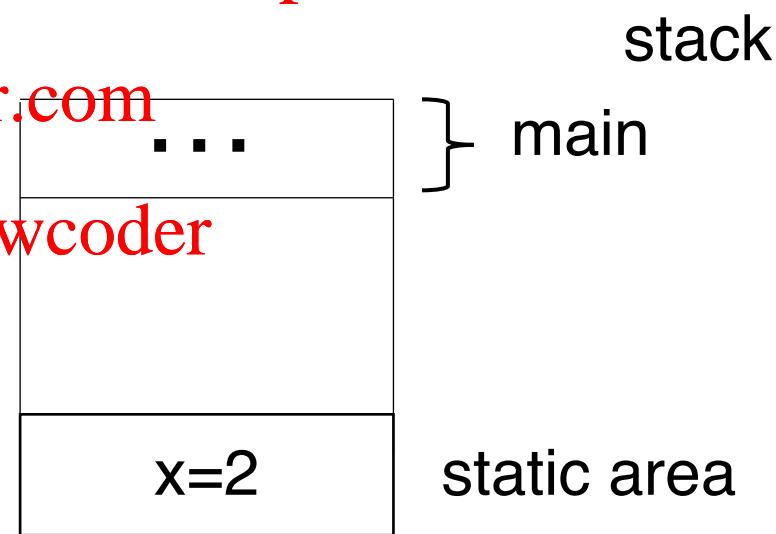
Call-By-Value

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
    print(x); //result?  
}
```



Call-By-Value

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
print(x); //result?  
}
```



Call-By-Value

Formal & Actual parameters have separate memory: their values may diverge

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Characteristics <https://powcoder.com>

- Actual parameters may not directly be changed in callee (unless pass in pointers)
- Arguments can be complex expressions
- Simple and intuitive (less error-prone)

Call-By-Value: Performance

copy value to actual parameter
to array

Primitive types: cost per parameter is small

Arrays, records, structures: copying value could be slow

<https://powcoder.com>

C: programme ~~Add WeChat pass pointers~~ pointers/references to avoid copying cost

Java: only primitive types use call-by-value

Call-By-Value-Return (In-Out)

Calling mechanism

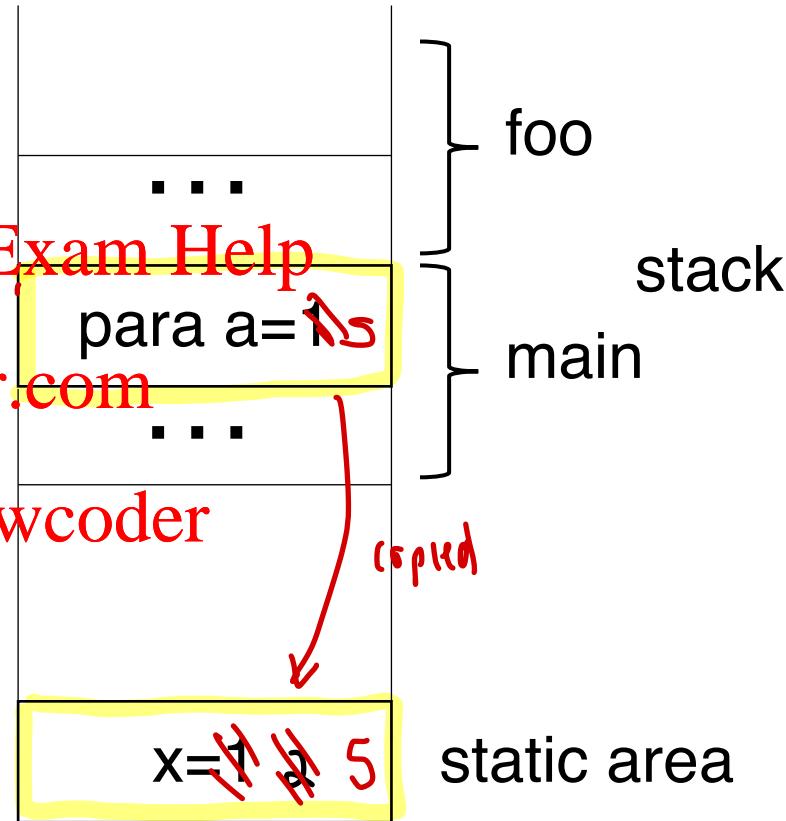
- Arguments are evaluated to their values
- Memory or registers allocated for arguments on AR
- Argument values stored in AR
- Before callee returns, AR values copied back to actual arguments
- AR destroyed when callee returns

- ① actual and formal parameters still have separate memory
 ② they become identical after callee returns

Call-By-Value-Return

```

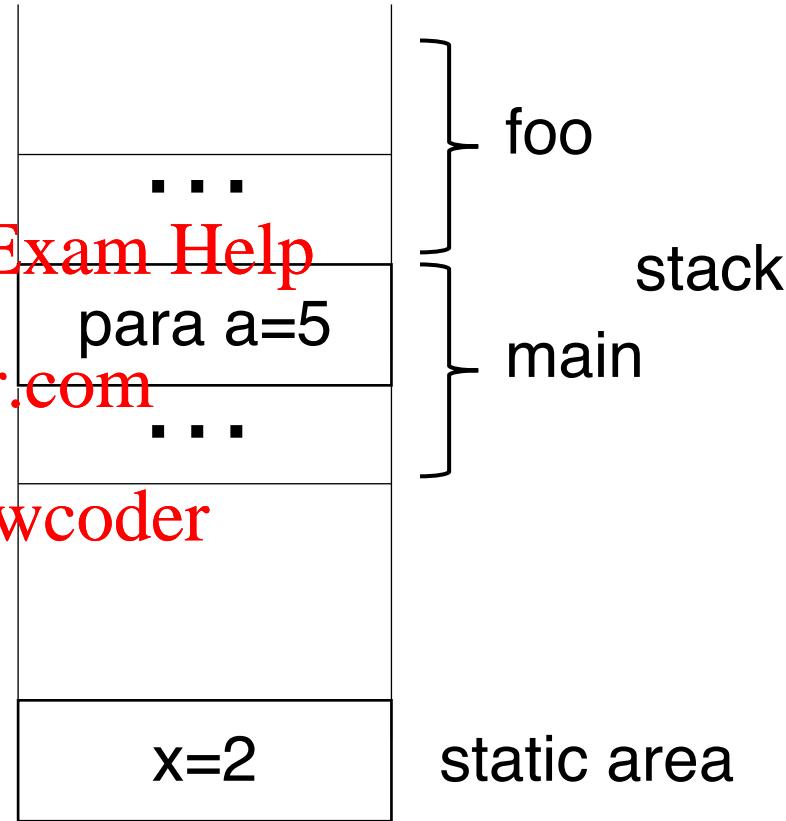
int x=1;
int foo (int a) {
    x = 2; Assignment Project Exam Help
    → print (a);
    5 → a = 5; https://powcoder.com
    7 → return x+a;
}
void main() {
    foo(x); //result? 7
    print(x); //result? 2
}
    
```



((BV))
5
((BVR))

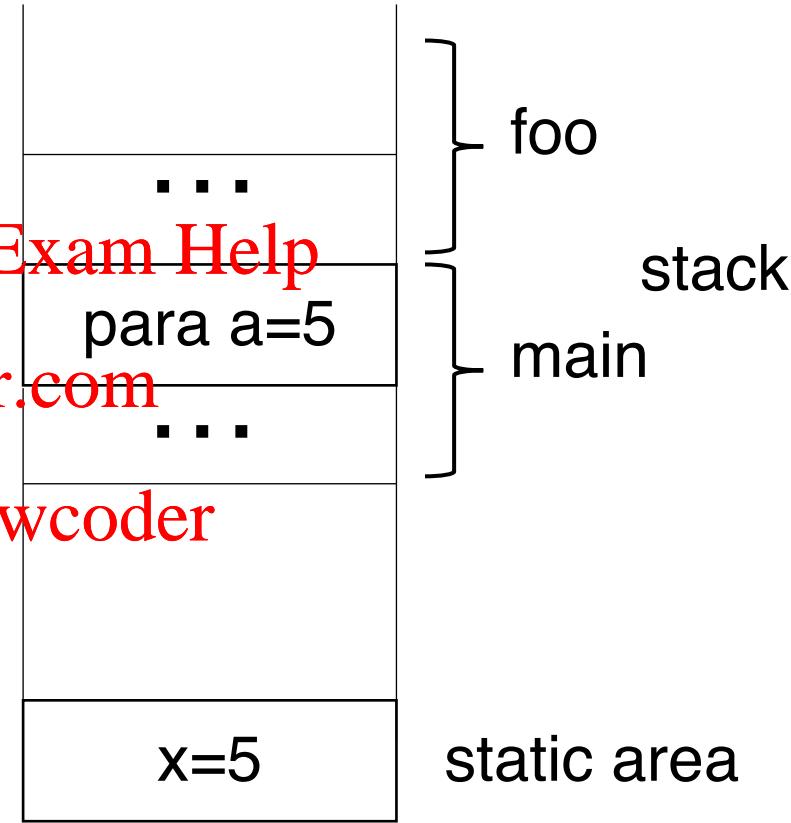
Call-By-Value-Return

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
    print (x); //result?  
}
```



Call-By-Value-Return

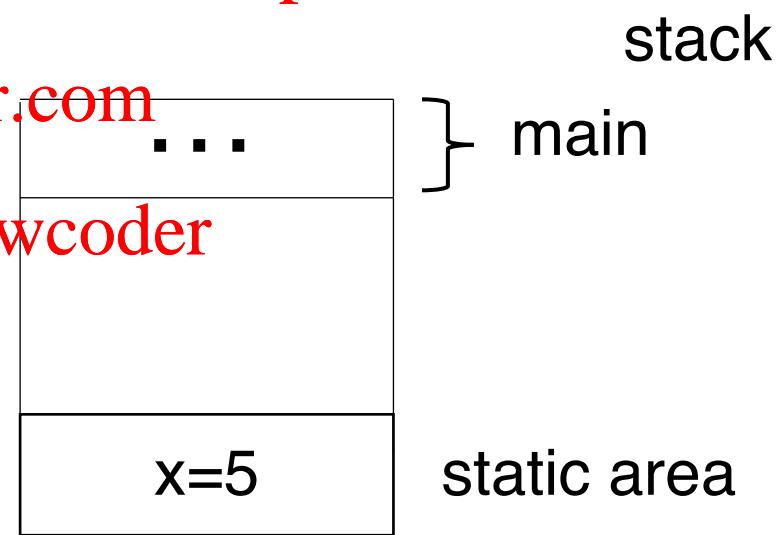
```
int x=1;  
int foo (int a) {  
    x = 2; //Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
    print(x); //result?  
}
```



AR values copied back
to actual arguments

Call-By-Value-Return

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
print(x); //result?  
}
```



Call-By-Value-Return

Characteristics

- Mostly identical to call-by-value, except an extra step of copying values back to actual parameter

[Assignment Project Exam Help
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

Call-By-Reference

Calling mechanism

- Arguments are evaluated to their values
- Memory or registers allocated for arguments on AR
- Argument **address** stored in AR
- AR destroyed when callee returns

Assignment Project Exam Help

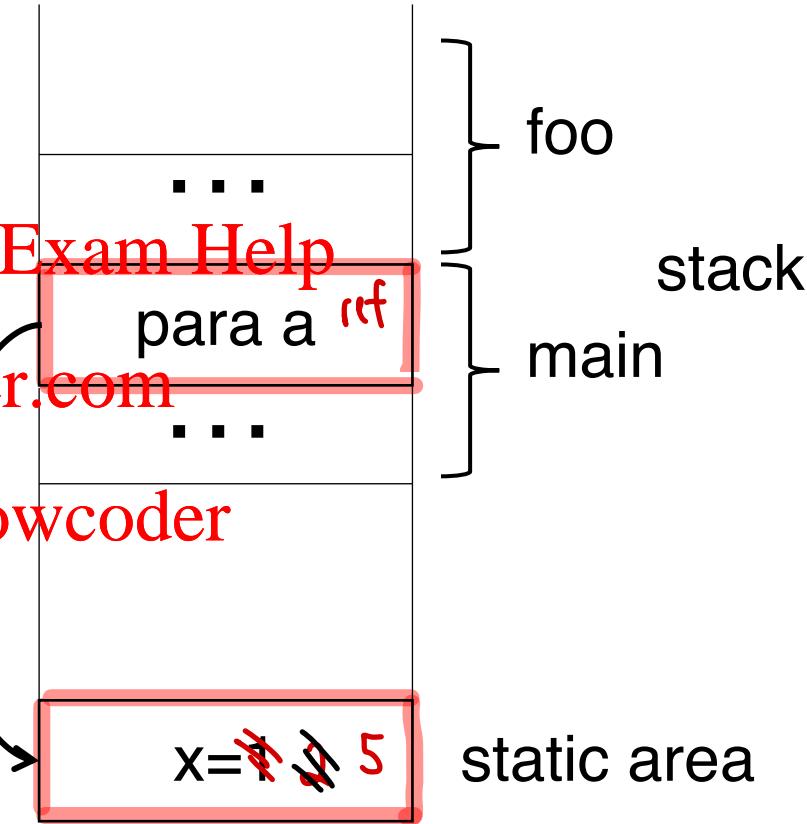
<https://powcoder.com>

Add WeChat powcoder

Call-By-Reference

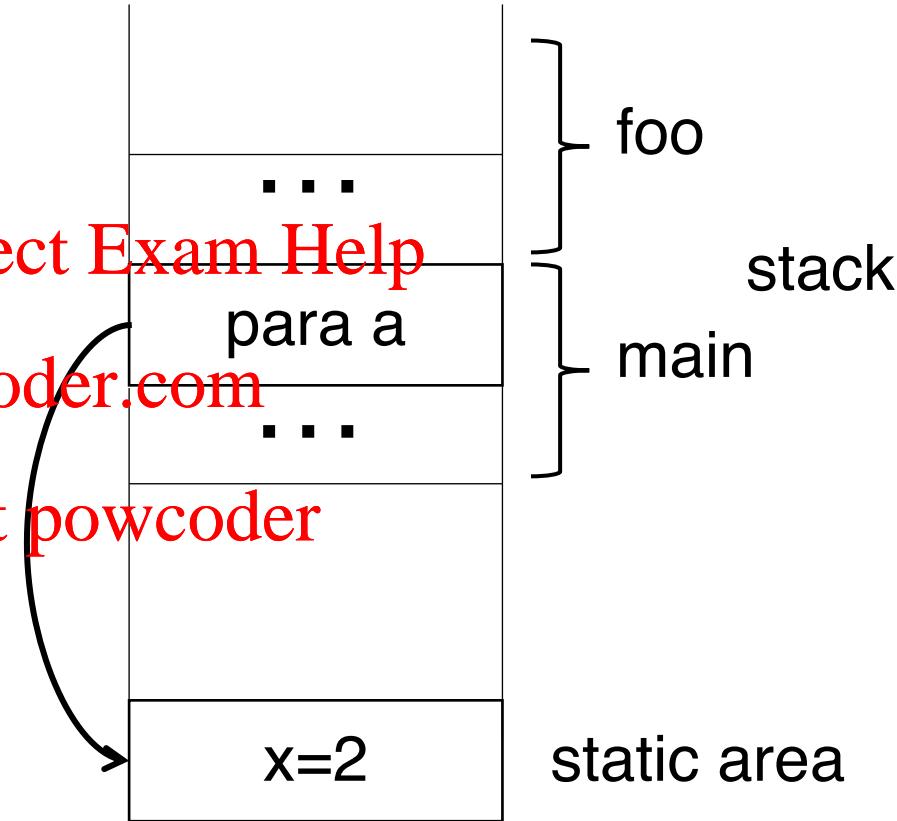
- ① actual and formal parameter refer to same memory
- ② they always have the same value

```
int x=1;           ↗ alias of x
int foo (int a) {
    → x = 2; Assignment
    → print (a); Project Exam Help
    → a = 5; https://powcoder.com
    return x+a; you update x
}
void main() {
    foo(x); //result? 10
    print(x); //result? 5
}
```



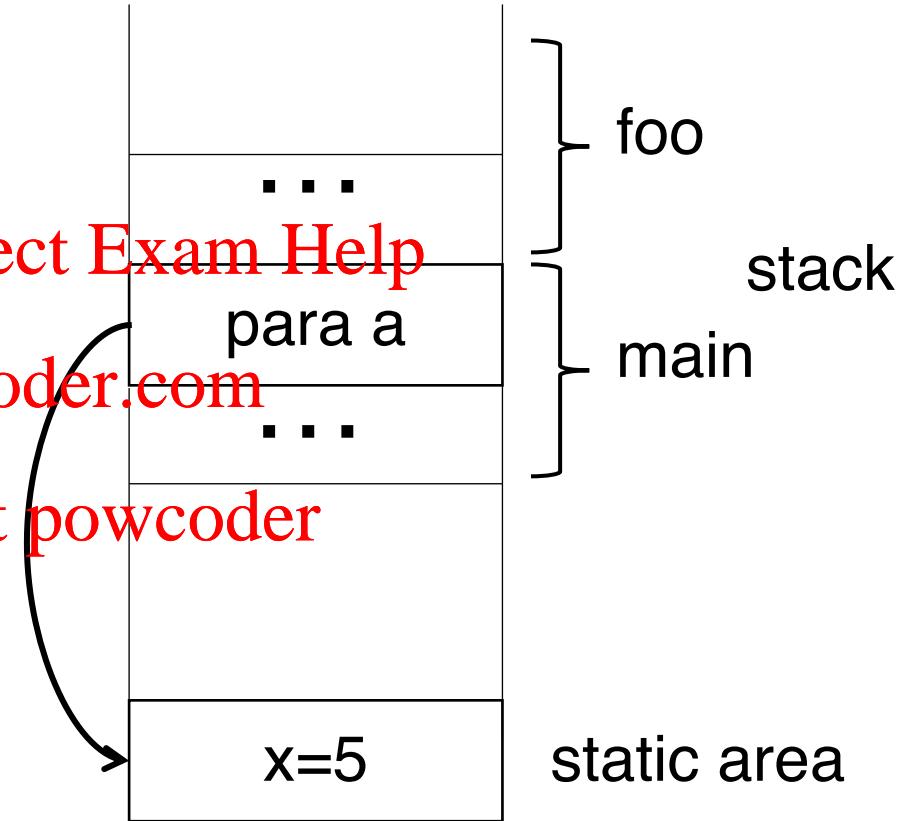
Call-By-Reference

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
    print (x); //result?  
}
```



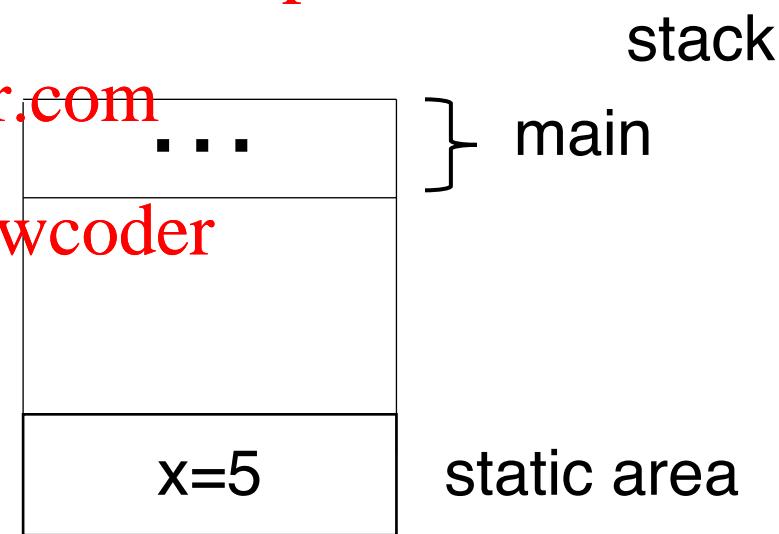
Call-By-Reference

```
int x=1;
int foo (int a) {
    x = 2; Assignment Project Exam Help
    print (a);
    a = 5; https://powcoder.com
    return x+a; Add WeChat powcoder
}
void main() {
    foo(x); //result?
    print (x); //result?
}
```



Call-By-Reference

```
int x=1;  
int foo (int a) {  
    x = 2; Assignment Project Exam Help  
    print (a); https://powcoder.com  
    a = 5; Add WeChat powcoder  
    return x+a;  
}  
void main() {  
    foo(x); //result?  
print(x); //result?  
}
```



Call-By-Reference

Formal parameter is an alias of actual parameter:
their values are the same

Assignment Project Exam Help

Characteristics <https://powcoder.com>

- Actual parameters may directly be changed in callee
- Some language disallows complex expressions as arguments
- Programs are harder to understand (more error-prone)

Call-By-Reference: Performance

Avoids the cost of memory copy

Indirect memory access: address → value
[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>
C: call-by-value is the default mode

[Add WeChat powcoder](#)
Java: constructed types use call-by-reference

#24

Assignment Project Exam Help
Procedures and Functions
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Parameter Modes

```
int p = 5;  
int q = 2;  
int f (int b, int c) {  
    b = 2 * c;  
    c = 3 + p;  
    return b+c;  
}
```

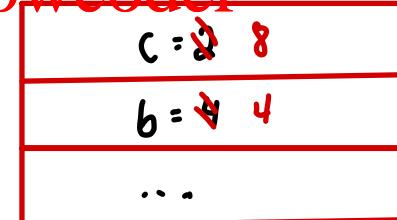
```
print f(p,q);  
print p;  
print q;
```

Outputs with
CBV? 1,5,1
CBVR? 1,4,1
CBR?

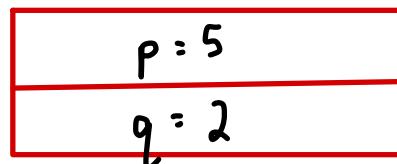
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



↑ stack



static
area

Parameter Modes

```
int p = 5;  
int q = 2;  
int f (int b, int c) {  
    b = 2 + p;  
    c = 3 + p;  
    return b+c;  
}  
print f(p,q);  
print p;  
print q;
```

Outputs with

CBV?

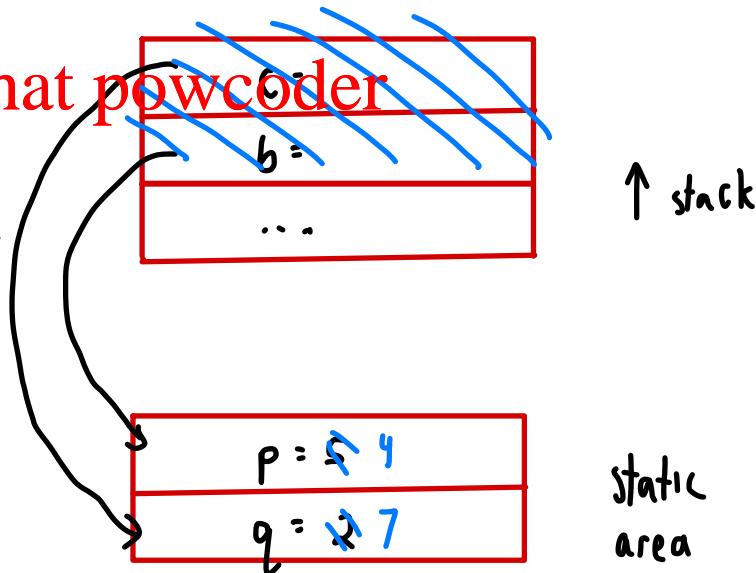
CBVR?

CBR? ||, ||

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Key Points

Call-by-value: Formal & Actual parameters have separate memory: their values may diverge

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Call-by-value-return: CBV+ copy final values of Formal parameters to Actual parameters

[Add WeChat](#) [powcoder](#)

Call-by-reference: Formal parameter is an alias of Actual parameter; their values are the same

Call-By-Name (Lazy Evaluation)

Calling mechanism

- Arguments are **not evaluated to their values** (like macros)
- Actual parameters replace all formal parameters in body

A
2,3,4

void swap (int a, int b) {
 int t = a;
 a = b;
 b = t;
}

swap (i, A[i]) // value swapped?
(2, A[2])₄

AddWeChat powcoder
https://powcoder.com
P A[i]

$t_1 \rightarrow 2$
 $i \rightarrow A[i]; \quad i \rightarrow 4$
 $A[i] \rightarrow t; \quad A[4] \rightarrow 2$
does not exist
exit

Still, a useful mode in functional programming, e.g., Haskell

Exceptions

So far, we assumed each function will run from start to its return points

Assignment Project Exam Help

But a program also needs to handle exceptions:
<https://powcoder.com>

- Out of memory
- Divide by zero
- File not found
- ...

Workaround in C language

- Return a special value (e.g., NULL) when a real value cannot be computed
- Return an explicit "status" to indicate if an exception happens <https://powcoder.com>
- Rely on the caller to pass in the exception handling code (e.g., via function pointer)

Exception handling is not enforced (error-prone)

Clutter up the program, especially for normal cases

Exception Handling

A language feature that

- Isolates error-checking code
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Direct execution to a handler when appropriate
<https://powcoder.com>

Define exceptions?
[Add WeChat](#) [powcoder](#)

How does exception change control flow?

Defining Exceptions

Pre-defined exceptions

- e.g., divide by zero, out-of-bound array access

Assignment Project Exam Help

<https://powcoder.com>

User-defined exceptions

- Typically, defined as a special kind of class

```
public class myException extends Exception
{ private int i;
  public myException(int x) {
    this.i = x; }
  public int getI() { return i; }
}
```

Throwing Exceptions

Pre-defined exceptions

- Automatically generated

Assignment Project Exam Help

<https://powcoder.com>

User-defined exceptions

- Generated by “throw” keyword

```
int foo ()  
{  
    ...  
    throw new myException(10);  
    ...  
}
```

Catching Exceptions

```
try
{
    .. code that might throw exceptions ..
}
catch (Exception1 e1) { .. handling code .. }
catch (Exception2 e2) { .. handling code .. }
...
finally { .. this code always executes,
         e.g. cleanup routine .. }
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

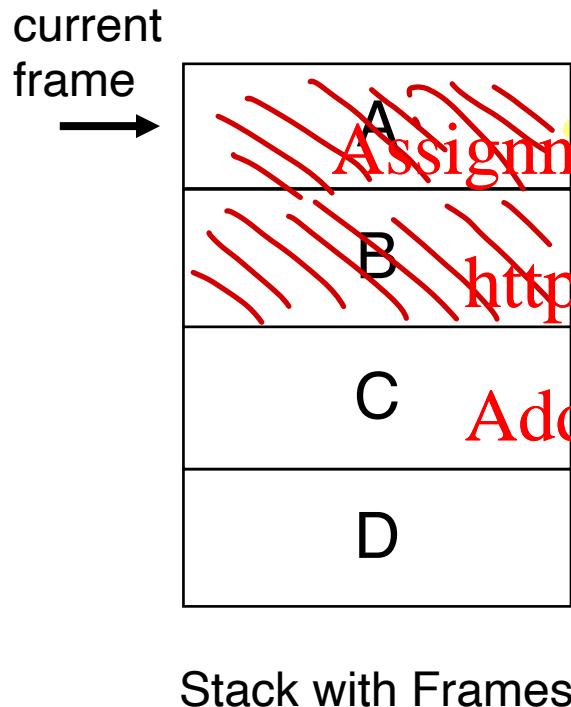
The “Replacement” Semantics

```
int foo () {  
    ...  
    try {  
        ...  
        throw new myException(10);  
        ... // this is replaced with handling code  
    } catch (myException e) {  
        ... Handling code ...  
    }  
    ...  
}
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder
throws level handling



Exception Propagation



If no handler is found

https://powcoder.com

If no handler is found

control flow continues after the handler in C

The exception mechanism needs to properly follow calling sequence to properly “exit” functions (e.g., restore registers, destroy frame)

Exception Implementation

When encounter an exception at pc:

- Go through handlers associated with pc in sequence
- Each handler either executes (when exception matches) or re-throws the exception for next handler
- If exception not handled, use *a function-level handler*: exit the function properly and re-throw the exception
- Now, pc is in the caller, and repeat the first bullet in the caller

#25

Assignment Project Exam Help

Types <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Values and Types

Values are type-less in hardware

0100 0000 0101 1000 0000 0000 0000 0000

Assignment Project Exam Help

Floating point number: 3.375 Two 16-bit integer: 16472 and 0
32-bit integer : 1,079,508,992 Four ASCII characters: @ X NUL NUL

Add WeChat powcoder

Operations expect certain values

- add take integers, fadd take floats

How to ensure operators get right *type* of values?

What Can Go Wrong?

Operation on wrong values will produce garbage

- float addition on two integers
- assign a string to a float
- pass an int to a function expecting a string

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

All of these errors are ***type errors***

Type

An abstraction of a set of values, and legal operations on these values.

Assignment Project Exam Help

- int: - 2^{31} to $2^{31}-1$, with operations +, -, *, /, ...

<https://powcoder.com>

Meaning of type Add WeChat powcoder

- Denotational: a set of value
- Constructive: set of primitive types, type constructors
- Abstraction: an interface (set of operations)

Why Types?

- Identify/Prevent type errors
- Program organization/abstraction
- Documentation
- Support optimization

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Kinds of Types

Primitive

Assignment Project Exam Help

Constructed

- Products
- Unions
- Arrays
- Lists

<https://powcoder.com>

Add WeChat powcoder

User-Defined

Type System

A method or specification for associating types with variables, expressions, etc.

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Type equivalence: are two types equivalent
<https://powcoder.com>

Add WeChat powcoder

Type compatibility: can int be used as float?

Type safety: absence of type errors

Type Checking

Strongly typed: all type errors caught by type checking

Weakly typed: type checking may miss type errors
[Assignment Project Exam Help](https://powcoder.com)

Static typing: type checking happens at compile time
<https://powcoder.com>

Dynamic typing: type checking happens at run time

Type inference: the type-checker infers types for variables
[Add WeChat powcoder](#)

```
union U {int a; float p} u;  
float x = 1.0;  
u.a = 1;  
x = x + u.p;
```

C is weakly typed, at compile time

Basic Types

Numeric

- bytes, integers, floats

Assignment Project Exam Help

Booleans

<https://powcoder.com>

Add WeChat powcoder

Characters

Data types available on contemporary machines

Integers

Length depends on language and compiler

Representation: two's complement format

Assignment Project Exam Help

n	0	binary representation of n
---	---	----------------------------

15	0	000 0000 0000 0000 0000 0000 0000 1111
----	---	--

-n	1	flip all bits of binary representation of n, and add 1
----	---	--

-5	1	111 1111 1111 1111 1111 1111 1111 1011
----	---	--

15-5?	0	000 0000 0000 0000 0000 0000 0000 1010
-------	---	--

Just add
binaries of
15 and -5

Floats

Single precision (float): 32 bits

Double precision (double): 64bits

Assignment Project Exam Help

<https://powcoder.com>

Due to the limited space, floats are *estimations* of the number
Add We Chat powcoder

```
float z = 1.345+1.123;  
printf("%d\n", z==2.468);
```

Floats

IEEE 754 Standard

Representation of floating point numbers in IEEE 754 standard:

Assignment Project Exam Help

single precision

sign	S	E	M
------	---	---	---

<https://powcoder.com>

exponent:

bias 127

mantissa:

sign + magnitude, normalized
binary integer binary significand w/ hidden
integer bit: 1.M

actual exponent is
 $e = E - 127$

Add WeChat powcoder

$$0 < E < 255$$

$$N = (-1)^S \cdot 2^{E-127} \cdot (1.M)$$

Boolean

Most languages: true or false

C: 0 means false, all other values mean true

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

In most implementations, a boolean value occupies more than one bit in memory (word is the basic unit of load/store)

[Add WeChat](#) [powcoder](#)

Character

All languages support ASCII code (7-bit)

Assignment Project Exam Help

Most modern language support Unicode (e.g.,
Java char uses <https://powcoder.com>, a 16-bit char set)

Add WeChat powcoder

Enumeration Types

Provide names to a sequence of integral values

C/C++

Assignment Project Exam Help

```
enum day {Monday, Tuesday, Wednesday,  
          Thursday, Friday, Saturday, Sunday};  
enum day myDay = Friday;
```

Add WeChat powcoder

Enumeration type improves readability

Enumeration Types

Provide names to a sequence of integral values

C/C++

Assignment Project Exam Help

```
enum day {Monday, Tuesday, Wednesday,  
          Thursday, Friday, Saturday, Sunday};  
enum day myDay = Friday;
```

Add WeChat powcoder

Java

```
enum Day {Monday, Tuesday, Wednesday,  
          Thursday, Friday, Saturday, Sunday};  
for (Day d: Day.values()) {  
    System.out.println(d);  
}
```

#26

Assignment Project Exam Help

Types <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Kinds of Types

Primitive

Assignment Project Exam Help

Constructed

- Products
- Unions
- Arrays
- Lists

<https://powcoder.com>

Add WeChat powcoder

User-Defined

Records and Structures

Usually laid out contiguously

Assignment Project Exam Help

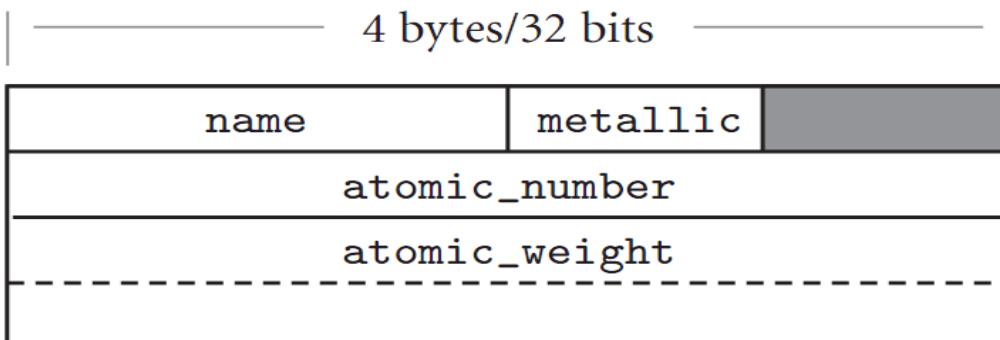
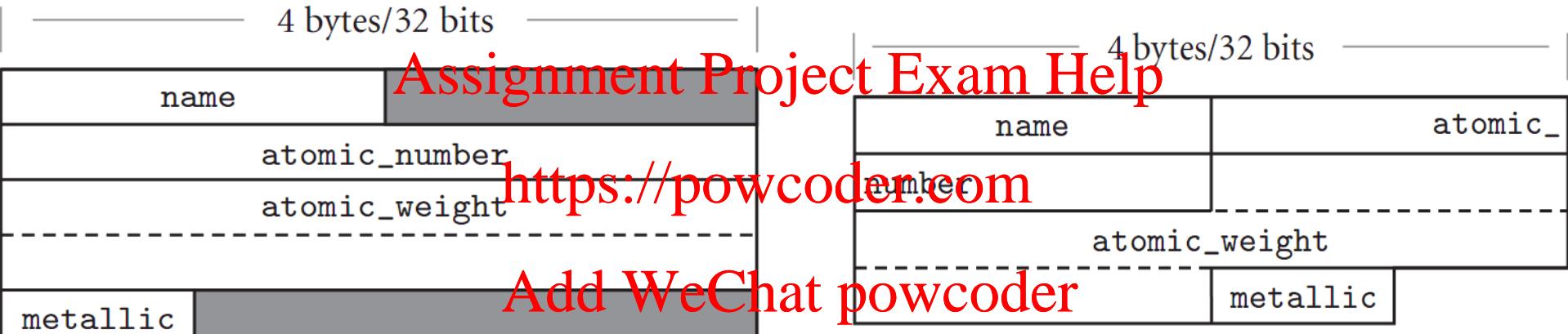
Possible holes for alignment

<https://powcoder.com>

```
struct id {  
    int i;  
    double d; } ;  
struct id x;  
x.i, x.d
```

Add WeChat powcoder
Compilers may re-arrange fields to minimize holes

```
struct element {  
    char name[2];  
    int atomic_number;  
    double atomic_weight;  
    _Bool metallic; }
```



Possible
Memory Layouts

Records and Structures

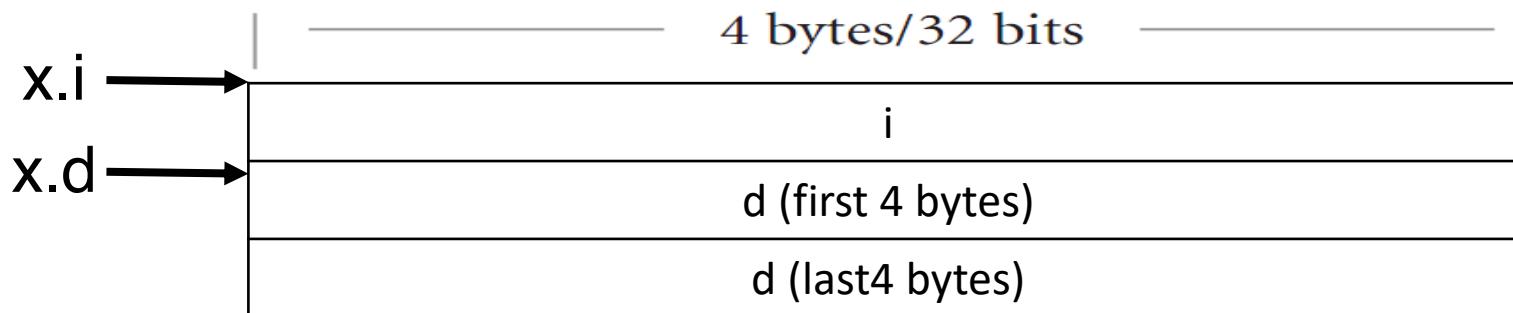
Usually laid out contiguously

Assignment Project Exam Help

<https://powcoder.com>

```
struct id {  
    int i;  
    double d; } ;  
struct id x;  
x.i, x.d
```

A possible memory layout: Add WeChat powcoder



Each field has a separate piece of memory

(Free) Union Types

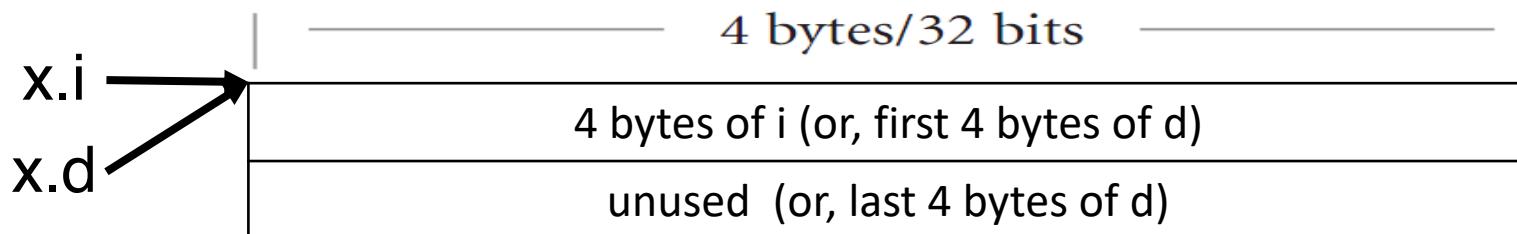
Laid out in shared memory

Assignment Project Exam Help

<https://powcoder.com>

A possible memory layout: Add WeChat powcoder

```
union id {  
    int i;  
    double d; } ;  
union id x;  
x.i = 1;  
y = 1.0 + x.d;
```



All fields share the same piece of memory

(Free) Union Types

Not type safe:

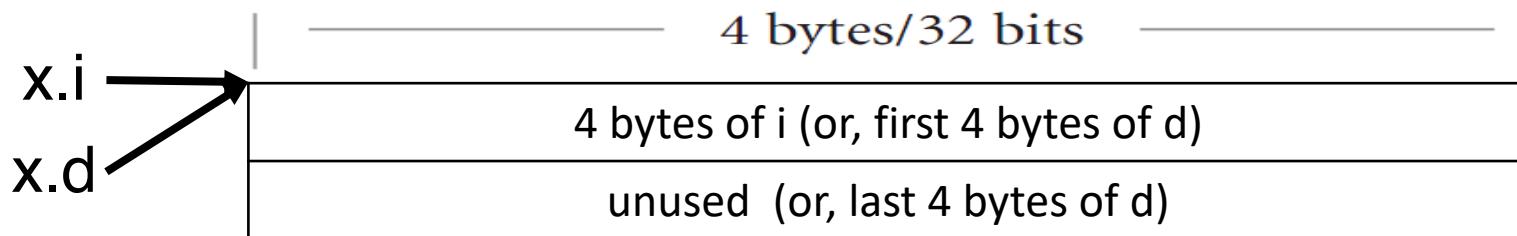
x.d will read the binary

0x00000001,???????? as a double

<https://powcoder.com>

```
union id {  
    int i;  
    double d; } ;  
union id x;  
x.i = 1;  
y = 1.0 + x.d;
```

A possible memory layout: [Add WeChat powcoder](#)



How can we make it type safe?

Discriminated Union Types

A combination of a tag (like an enum) and a payload per possibility (like a union).

Assignment Project Exam Help

```
enum Kind {isInt, isFloat}  
struct intOrReal {  
    enum Kind which;  
    union U {int a; float p} u;  
} ir;  
float x = 1.0;  
if (ir.which == isInt) ir.u.a = 1;  
if (ir.which == isFloat) x = x + ir.u.p;
```

Still not type safe: type system doesn't enforce tag check

Sum Types

Many functional programming languages support type-safe sum types

[Assignment](#) [Project](#) [Exam](#) [Help](#) (possibly empty)

Haskell

Tag
<https://powcoder.com>
payload type

```
data intorreal = isInt Int | isFloat Float
-- given u has type intorreal
case u of
  isInt i -> i + 1
  isFloat f -> f + 1.0
```

Type safe: type is checked under each case statement (the only way to read from a value with the sum type)

Sum Types are General

Haskell

Tag

```
data day = Monday | Tuesday | Wednesday |
          Thursday | Friday | Saturday | Sunday
-- Given d has type day
case d of
    Monday -> ... Add WeChat powcoder
    Tuesday -> ...
    ...
    ...
```

Assignment Project Exam Help

<https://powcoder.com>

A generalization of Enumeration type

Sum Types and Product Types

Sum Types: alternation of types

Product Types: concatenation of types (such as?)

(records and structures are product types)

<https://powcoder.com>

```
enum Color {Red, Blue}  
enum Shape {Circle, Rectangle}  
struct ColoredShape {enum color c; enum shape s}
```

Analogy:

Values
of color

Values
of shape

Values of
coloredShape

$$(Red + Blue) * (Circle + Rectangle)$$

$$= (Red * Circle) + (Red * Rectangle) + (Blue * Circle) + (Blue * Rectangle)$$

Array

Lifetime and array size

- Global lifetime, Static shape

Assignment Project Exam Help

- Local lifetime, Static shape

<https://powcoder.com>

Add WeChat powcoder

- Local lifetime, Dynamic shape

```
int f() {  
    int A[10];  
}
```

```
int f(int n) {  
    int A[n];  
    ... }
```

Bounds Checking

Is it done?

Assignment Project Exam Help

<https://powcoder.com>

How is it done?

Add WeChat powcoder

When is it done?

```
int f() {  
    int A[10];
```

...

...

```
int f(int n) {  
    int A[n];
```

...

A[e]

...

}

C Array

Static/Stack/Heap allocated

Size statically/dynamically determined
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Array bounds not checked (buffer overflow)
<https://powcoder.com>

Add WeChat powcoder

Java Array

Heap allocated

Size dynamically determined

Assignment Project Exam Help

Array size is part of stored data (Dope Vector)

<https://powcoder.com>

Array bounds checked

Add WeChat powcoder

#27

Assignment Project Exam Help

Types <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

C Array

Static/Stack/Heap allocated

Size statically/dynamically determined
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Array bounds not checked (buffer overflow)
<https://powcoder.com>

Add WeChat powcoder

Java Array

Heap allocated

Size dynamically determined

Assignment Project Exam Help

Array size is part of stored data (Dope Vector)

<https://powcoder.com>

Array bounds checked

Add WeChat powcoder

Dope Vectors (one example)

4	int	10	A[0]	A[1]	...	A[9]
---	-----	----	------	------	-----	------

elem. elem. array
size type size

Assignment Project Exam Help
<https://powcoder.com>
Pointer of A

Add WeChat powcoder

Address of A[i]?

$A + 4 * i$

Bound check?

$0 \leq i < 10$

Benefit: the array may change dynamically

Stack vs Heap Allocation

Size statically determined; fixed

- Allocation in the usual way

Assignment Project Exam Help

Size dynamically determined; fixed

- May still be allocated in the stack
- Need shape information at run time (dope vector)

Size may change

- Heap allocated

Memory Layout

One-dimensional arrays

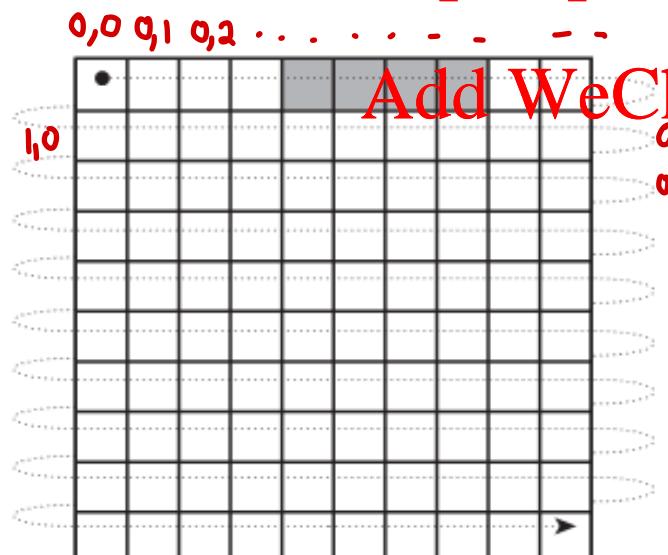


Assignment Project Exam Help

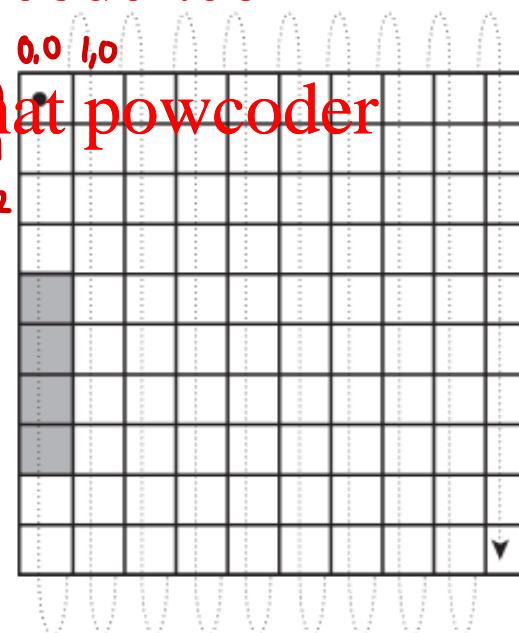
Two-dimensional arrays

```
int[][] A = new int[10][100]
```

<https://powcoder.com>



Row-major order



Column-major order

Memory Layout

Row-Pointer Layout

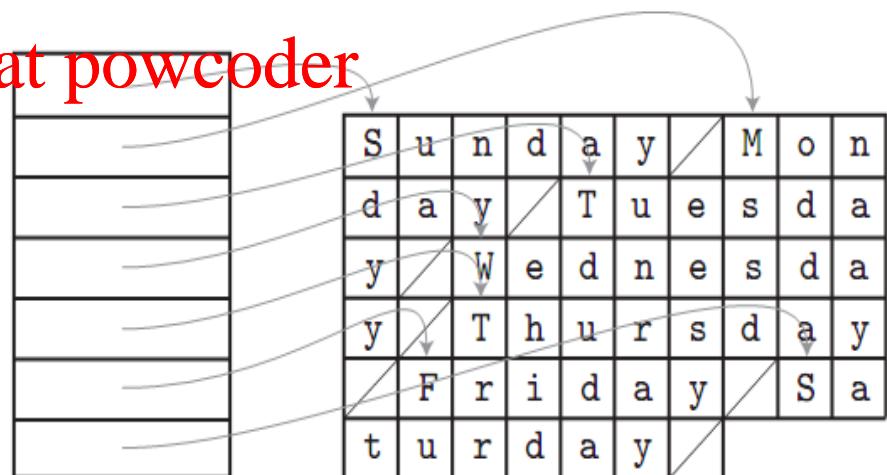
```
int[][] B = new int[10][]
B[0] = new int[100]
B[1] = new int[50]
```

Assignment Project Exam Help

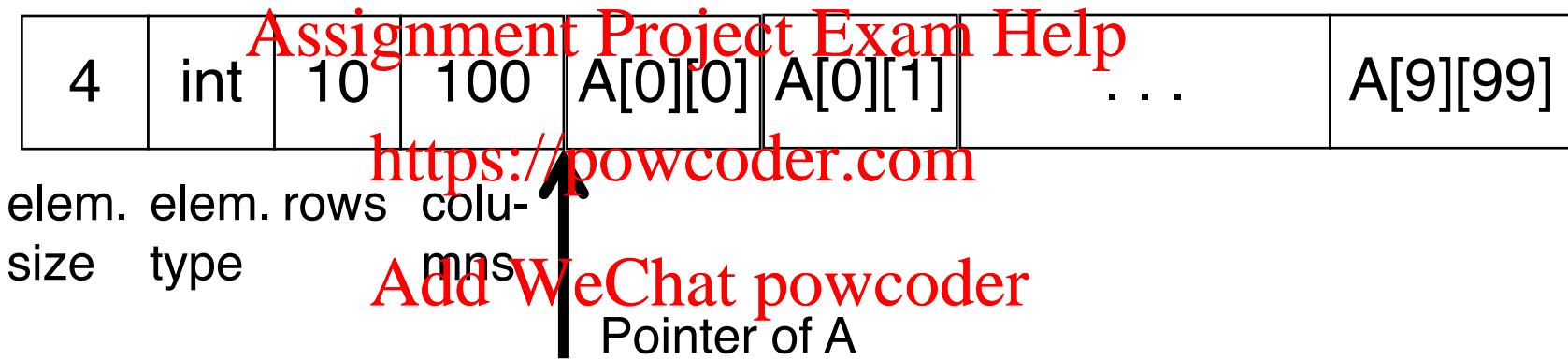
<https://powcoder.com>

S	u	n	d	a	y					
M	o	n	d	a	y					
T	u	e	s	d	y					
W	e	d	n	e	s	d	a	y		
T	h	u	r	s	d	a	y			
F	r	i	d	a	y					
S	a	t	u	r	d	a	y			

Add WeChat powcoder



Address Calculation (Row major)



Address of $A[i][j]$?
Bound check?

$A + 4(i * 100 + j)$
 $0 \leq i < 10, 0 \leq j < 100$

Address Calculation (Column major)

4	int	10	100	A[0][0]	A[1][0]	...	A[9][99]
elem. size	elem. type	rows	cols	https://powcoder.com			
mns				Add WeChat powcoder Pointer of A			

Address of A[i][j]?
Bound check?

$$\begin{aligned} \text{Address of } A[i][j] &? & A + 4(j*10+i) \\ \text{Bound check?} & & 0 \leq i < 10, 0 \leq j < 100 \end{aligned}$$

Pointers

What are they?

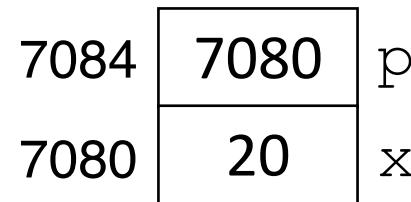
- A set of memory addresses and operations on them

Assignment Project Exam Help

Values: legal addresses, and a special value, nil

Add WeChat powcoder
address

```
int x=20;  
int* p = &x;
```



Pointers

Operations: assignment, dereferencing, arithmetic

```
int x=20;  
int* p = &x;  
int y=*p;
```

Assignment Project Exam Help
int a[3]={1,2,3};
int x = *(a+1) //same as a[1]
<https://powcoder.com>

Add WeChat powcoder

Uses

- Indirect addressing (access arbitrary address)
- Manage dynamic storage (heap)

Pointers vs. References

Pointers: int *p;

References: int &p;

<https://powcoder.com>

Value Model vs. Reference Model: A = B

- Value model: the value of B is copied to A
- Reference model: A is an alias of B (same memory)
- Java: primitive types follow value model; objects follow reference model

References

Restricted pointers: cannot be used as value or operated in any way

Assignment Project Exam Help

Not directly visible to the programmer

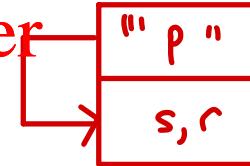
No explicit data type

double r=2.3;

double& s=r; //s is an alias of r (share memory)

double *p = &r; //p has value: address of r

s += 1; *p += 1;



References

Uses

- ~~Indirect addressing (access arbitrary address)~~
Assignment Project Exam Help
- Manage dynamic storage (heap)
<https://powcoder.com>

Alias of existing **Add WeChat** powcoder

The Nil Pointer Problem

"I call it my billion-dollar mistake. It was the invention of the null reference in 1965. ... This has led to innumerable errors, vulnerabilities, ~~Assignment Project Exam Help~~, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years."

<https://powcoder.com>

C.A.R. Hoare, 2009
Add WeChat powcoder

How to avoid it?

The Nil Pointer Problem

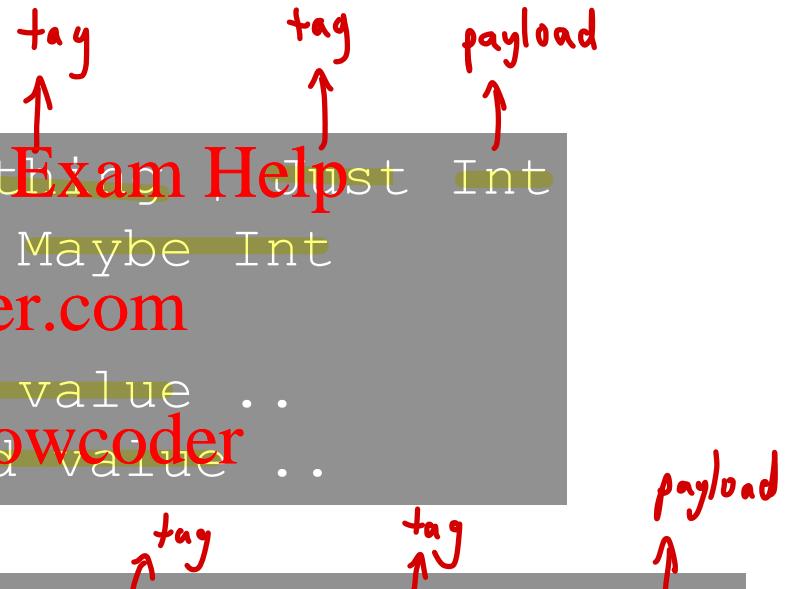
Solution: Use sum types

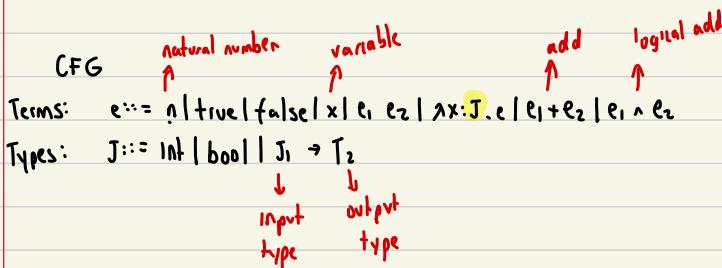
Haskell

```
data Maybe a = Nothing | Just a  
-- given p has type Maybe Int  
case p of  
    Nothing => ... nil value ...  
    Just i => ... valid value ...
```

SML

```
datatype int option = NONE | SOME of int  
-- given p has type int option  
case p of  
    None => ... Nil value ...  
    | SOME i => ... Valid value ...
```



Simply Typed Lambda Calculus

example:

type system: takes a program as input and returns either

① $3 : \text{int}$
 or $\lambda x: \text{int}. x$ } type safe

the type of the program or rejects the program

② $\lambda x: \text{int}. (x+1)$

③ $(\text{true} 3)$ } type unsafe

true is boolean value not function

<https://powcoder.com>

$\Gamma \vdash n : \text{int}$ (T-NUM) $\Gamma \vdash \text{true} : \text{bool}$ (T-TRUE) $\Gamma \vdash \text{false} : \text{bool}$ (T-FALSE) $\Gamma, x: \text{int} \vdash x : \text{int}$ (T-VAR)

under any Γ and $x: \text{int}$
 number n has type int assumption conclusion

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash (e_1 + e_2) : \text{int}} \quad (\text{T-ADD})$$

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash (e_1 \vee e_2) : \text{bool}} \quad (\text{T-AND})$$

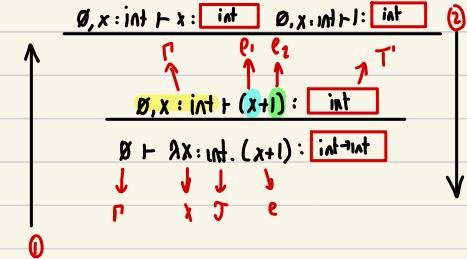
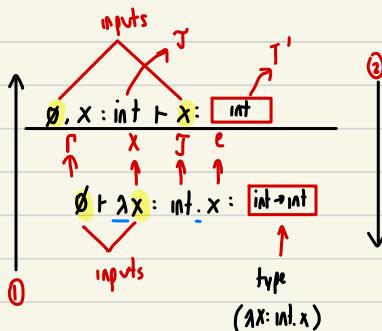
proves
 typing judgement has type
 $\Gamma \vdash e : J$ (Name)
 ↓
 typing environment
 program being checked has type
 type of e

$\Gamma \vdash e : Y$
 ✓ output
 inputs to type system

assumptions conclusions inference rule

$\Gamma \vdash n : \text{int}$
 $\Gamma \vdash \text{true} : \text{bool}$
 $\emptyset \vdash 3 : \text{int}$
 ↓
 empty symbol table
 inputs to type system

PROOF TREE



Assignment Project Exam Help

There is no way to interpret a boolean type as a function type
for any program there is a unique proof tree

$\emptyset \vdash \text{true} : \text{bool}$ $\emptyset \vdash 3 : \text{int}$

$\emptyset \vdash (\text{true} 3) : \text{int}$

X

cannot type check

<https://powcoder.com>

Add WeChat powcoder

1 Introduction

Type checking is a lightweight technique for proving simple properties of programs. Type checking usually cannot determine if a program will produce the correct output or not. Instead, it is a way to test whether a program is well-formed, with the idea that a well-formed program satisfies certain desirable properties. The traditional application of type checking is to show that a program cannot get stuck. In other words, a type-correct program will never reach a state from which the programs behavior is undefined (e.g., adding two floats by an integer addition). This is a weak notion of program correctness, but nevertheless very useful in practice for catching bugs. Type systems are a powerful technique. In the past couple of decades, researchers have discovered how to use type systems for a variety of different verification tasks, such as verifying information flow security at compile time.

2 A simply typed lambda calculus

To understand how type system works, we consider a typed variant of the λ -calculus we have seen earlier in this class. In this language, we assign types to certain λ -terms according to some typing rules. A λ -term is considered to be well-formed if a type can be derived for it using the rules. In this lecture, we will not formalize and prove the properties of this type system. But we will see the properties informally in the end.

Assignment Project Exam Help

2.1 Syntax

The language syntax is similar to that of untyped λ -calculus, with some notable differences. There are two kinds of inductively-defined expressions, terms and types. The definition of this language can be written in context-free grammar as follows. As standard, we use e_1 and e_2 to distinguish multiple uses of the same nonterminal e in the grammar.

terms	$e ::= n \mid \text{true} \mid \text{false} \mid x \mid e_1 e_2 \mid \lambda x : \tau . e \mid e_1 + e_2 \mid e_1 \wedge e_2$
types	$\tau ::= \text{int} \mid \text{bool} \mid \tau_1 \rightarrow \tau_2$

One difference from the pure λ -calculus is that the natural numbers (n) and Boolean constants (`true` and `false`) are taken to be primitive symbols (just as in C and Scheme). This language also contains primitive operations: the plus operation (+) on natural numbers, as well as the and operation (\wedge) on Boolean values. Another difference is that a λ -abstraction explicitly mentions the type of its argument (symbol τ in the term $\lambda x : \tau . e$ is the type of parameter x). A type τ in this language is either a primitive type (`int`, `bool`), or a constructed function type. Here, $\tau_1 \rightarrow \tau_2$ represents a function from type τ_1 to type τ_2 .

A *value* is either a number, a Boolean constant, or a closed λ -abstraction $\lambda x : \tau . e$. There is a set of typing rules, given below, that can be used to associate a type with a term. If a type τ can be derived for a term e according to the typing rules, we write $\vdash e : \tau$, read as “ e has type τ ”. This metaexpression is called a *type judgment*. For example, every number has type `int`, thus $\vdash 3 : \text{int}$. Boolean value `true` has type `bool`, thus $\vdash \text{true} : \text{bool}$. A function $\lambda x : \text{int} . \lambda y : \text{int} . x$ has the type $\text{int} \rightarrow (\text{int} \rightarrow \text{int})$. The \rightarrow constructor associates to the right, so $\text{int} \rightarrow (\text{int} \rightarrow \text{int})$ is the same as $\text{int} \rightarrow \text{int} \rightarrow \text{int}$. Thus we can write

$$\vdash (\lambda x : \text{int} . \lambda y : \text{int} . x) : \text{int} \rightarrow \text{int} \rightarrow \text{int}$$

Not all λ -terms can be typed, for instance $(\lambda x : \text{int} . x x)$ or $(\text{true} 3)$. These expressions are considered nonsensical. Intuitively, such terms are ill-formed. These terms should be rejected by a type system that ensures type-safety. Next, we will define a type system that ensures that any well-typed term will never get stuck. For example, both $(\lambda x : \text{int} . x x)$ and $(\text{true} 3)$ are rejected by the type system.

2.2 Typing rules

The typing rules will determine which terms are well-formed terms. They are a set of rules that allow the derivation of type judgments of the form $\Gamma \vdash e : \tau$, read as “e has type τ in context Γ ”.

Here Γ is a type environment, a partial map from variables to types used to determine the types of the free variables in e . The domain of Γ , written as $\text{dom}(\Gamma)$, is a subset of variables, whose types are defined in the type environment. The environment $\Gamma[\tau/x]$ is obtained by rebinding x to τ in Γ (or creating the binding anew if x is not in the domain of Γ):

$$\Gamma[\tau/x](y) \triangleq \begin{cases} \Gamma(y), & \text{if } y \neq x \text{ and } y \in \text{dom}(\Gamma) \\ \tau, & \text{if } y = x \\ \text{undefined}, & \text{otherwise} \end{cases}$$

We can also view type context Γ as a sequence of variables and their types. The notation $\Gamma, x : \tau$, which is equivalent to $\Gamma[\tau/x]$, follows this view. In this lecture, we also use $\Gamma, x : \tau$ to represent that τ is the most recent binding of x in the context. This notation is used frequently in the literature. Also, one often sees $x : \tau \in \Gamma$, which means just $\Gamma(x) = \tau$.

We also write $\Gamma \vdash e : \tau$ as a metaexpression to mean that the type judgment $\Gamma \vdash e : \tau$ is derivable from the typing rules. The environment \emptyset is the empty environment, and the judgment $\vdash e : \tau$ is short for $\emptyset \vdash e : \tau$. The typing rules are:

Assignment Project Exam Help

$\Gamma \vdash n : \text{int}$ (T-NUM) $\Gamma \vdash \text{true} : \text{bool}$ (T-TRUE) $\Gamma \vdash \text{false} : \text{bool}$ (T-FALSE) $\Gamma, x : \tau \vdash x : \tau$ (T-VAR)

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1 e_2 : \tau'} \text{ (T-APP)}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash (\lambda x : \tau . e) : \tau \rightarrow \tau'} \text{ (T-ABS)}$$

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash (e_1 + e_2) : \text{int}} \text{ (T-ADD)}$$

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash (e_1 \wedge e_2) : \text{bool}} \text{ (T-AND)}$$

The typing rules are presented as *inference rules*, which inductively define relations consisting of the derivation of types. Each typing rule has three parts: the possibly empty assumption (type judgments above the bar), the conclusion (the type judge below the bar), and a name for the typing rule (in parentheses). When the assumption is empty, we simply write the conclusion without the bar.

Now let's take a closer look at these typing rules:

- The first three rules just say that all the base values have their corresponding base types.
- (T-Var) For a variable x , x has a type τ if the binding $x : \tau$ appears as the most recent binding of x in the type environment.
- (T-App) An application expression $e_1 e_2$ represents the result of applying the function represented by e_1 to the argument represented by e_2 . For this to have type τ' , e_1 must be a function of type $\tau \rightarrow \tau'$ for some τ , and its argument e_2 must have type τ .
- (T-Abs) A λ -abstraction $(\lambda x : \tau . e)$ is supposed to represent a function. The type of the input should match the annotation in the term, thus the type of the function must be $\tau \rightarrow \tau'$ for some τ' . The type τ' of the result is the type of the body under the extra type assumption $x : \tau$.
- The last two rules say the operands of add (resp. and) must have type `int` (resp. `bool`), and the result has type `int` (resp. `bool`).

Every well-typed term has a *proof tree* consisting of applications of the typing rules to derive a type for the term. We can *type-check* a term by constructing this proof tree. For example, consider the term $\lambda x : \text{int} . (x + 1)$, which takes a natural number x , and returns $x + 1$, another natural number. So we expect $\vdash (\lambda x : \text{int} . (x + 1)) : \text{int} \rightarrow \text{int}$. Here is a proof of that fact:

$$\frac{\begin{array}{c} x : \text{int} \vdash x : \text{int} \quad x : \text{int} \vdash 1 : \text{int} \\ \hline x : \text{int} \vdash (x + 1) : \text{int} \end{array}}{\vdash (\lambda x : \text{int} . (x + 1)) : \text{int} \rightarrow \text{int}}$$

Type checking starts from the bottom judgement. It has an empty typing environment (there is no global variable), and the goal of type checking is to derive the type on the right (initially unknown). To derive the type, at each step, we use the typing rule that matches the term being typed. Here, we first apply rule T-Abs to check function body $x + 1$ under environment $x : \text{int}$. The rule application gives us the second to last judgement in the proof tree, whose type after ‘ $:$ ’ is still unknown at this moment. Next, we apply rule T-Add to type-check x and 1. Here, we can fill in the rightmost types, since axioms T-Var and T-Num apply, and we no longer need the assumptions to derive the correct type. With the complete assumption, we can finish the type of $x + 1$ using rule T-Add, and similarly, fill in the type in the last judgement to complete the proof tree. Deriving types using a proof tree typically involves this “bottom-up” and then “top-down” flavor. You can use the next proof trees as an excise.

Consider a more complex term $((\lambda x : \text{int} . \lambda y : \text{bool} . x) 2 \text{ true})$, which evaluates to true . Since $\vdash 2 : \text{int}$, we expect $\vdash ((\lambda x : \text{int} . \lambda y : \text{bool} . x) 2 \text{ true}) : \text{int}$ as well. Here is a proof of that fact:

$$\frac{\begin{array}{c} x : \text{int}, y : \text{bool} \vdash x : \text{int} \\ \hline x : \text{int} \vdash (\lambda y : \text{bool} . x) : \text{bool} \rightarrow \text{int} \end{array}}{\vdash (\lambda x : \text{int} . \lambda y : \text{bool} . x) : \text{int} \rightarrow \text{bool} \rightarrow \text{int} \quad \vdash 2 : \text{int}} \frac{}{\vdash ((\lambda x : \text{int} . \lambda y : \text{bool} . x) 2) : \text{bool} \rightarrow \text{int} \quad \vdash \text{true} : \text{bool}} \frac{}{\vdash ((\lambda x : \text{int} . \lambda y : \text{bool} . x) 2) \text{ true} : \text{int}}$$

An automated type checker can effectively construct proof trees like this in order to test whether a program is type-correct. A term *type-checks* if there is a proof tree for that term; otherwise, the term is rejected by the type system. Most type-systems are decidable, and in fact, efficient. The complexity of most type systems ensuring type safety (including the one in this lecture) is linear to program size.

Note that types, if they exist, are unique. That is, if $\Gamma \vdash e : \tau$ and $\Gamma \vdash e : \tau'$, then $\tau = \tau'$. This can be proved easily by structural induction on e , using the fact that there is exactly one typing rule that applies in each case, depending on the form of e .

3 Type Safety

The type system enforces a very important property. Informally, we can state the property as follows: if a term e has type τ in context Γ , and e is executed in a memory state that is consistent with Γ (e.g., x has value 1 if $\Gamma(x) = \text{int}$), the final value of e must be a value of type τ . One implication is that since the type system checks that all operands to an operator (e.g., $+$ and \wedge in our language) have the proper type, an operator can never get a value with wrong type at run time. Hence, any well-formed program is type safe. However, the formal definition and proof of this property are beyond the scope of this lecture.

Type Inference

assign type of x to α
 $\lambda x.e$

terms $e ::= n \mid \text{true} \mid \text{false} \mid x \mid e_1 e_2 \mid \lambda x : \tau . e \mid e_1 + e_2 \mid e_1 \wedge e_2$
types $\tau ::= \text{int} \mid \text{bool} \mid \tau_1 \rightarrow \tau_2$

$\lambda x : \cancel{\text{int}}. (x+1) \rightarrow \text{int}$
add constraint $\alpha = \text{int}$

$((\lambda x : \cancel{\text{int}}. \lambda y : \cancel{\text{bool}}. x) 2) \text{ true}$

$x+1 = 2 \Rightarrow x=1$
 $y-10 = 2 \Rightarrow y=12$ constraints solved

constraints w/ unknown

variables (λ) Add WeChat powcoder

- 1) declare type constraints from program
- 2) solve type constraints

Type Constraints

$C ::= C_1 ; C_2 \mid \tau_1 = \tau_2$

$\alpha = \text{int}$

$r ::= \alpha / \text{int} / \text{bool} \mid r_1 \rightarrow r_2$

$\text{int} = \text{int} ; \text{bool} \neq \text{int}$

$\Gamma \vdash n : \text{int} | \emptyset \text{ (T-NUM)} \quad \Gamma \vdash \text{true} : \text{bool} | \emptyset \text{ (T-TRUE)} \quad \Gamma \vdash \text{false} : \text{bool} | \emptyset \text{ (T-FALSE)}$

$$\frac{\Gamma, x : \tau \vdash x : \tau | \emptyset \text{ (T-VAR)}}{\Gamma \vdash e_1 : \tau \rightarrow \tau_1 | C_1 \quad \Gamma \vdash e_2 : \tau_2 | C_2} \frac{}{\Gamma \vdash e_1 e_2 : \tau_1 | C_1; C_2; \tau = \tau_2} \text{ (T-APP)}$$

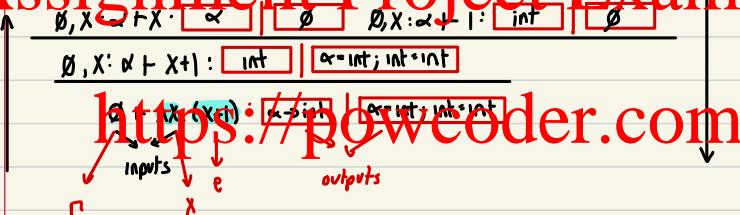
$$\frac{\Gamma, x : \tau \vdash e : \tau' | C}{\Gamma \vdash (\lambda x : \tau . e) : \tau \rightarrow \tau' | C} \text{ (T-ABS1)} \quad \frac{\Gamma, x : \alpha \overset{\text{int}}{\circlearrowleft} e : \tau | C \quad \alpha_x \text{ a fresh variable}}{\Gamma \vdash (\lambda x . e) : \alpha_x \rightarrow \tau' | C} \text{ (T-ABS2)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 | C_1 \quad \Gamma \vdash e_2 : \tau_2 | C_2}{\Gamma \vdash (e_1 + e_2) : \text{int} | C_1; C_2; \tau_1 = \text{int}; \tau_2 = \text{int}} \text{ (T-ADD)}$$

pre: $\Gamma | C$ constraints

$$\frac{\Gamma \vdash e_1 : \tau_1 | C_1 \quad \Gamma \vdash e_2 : \tau_2 | C_2}{\Gamma \vdash (e_1 \wedge e_2) : \text{bool} | C_1; C_2; \tau_1 = \text{bool}; \tau_2 = \text{bool}} \text{ (T-AND)}$$

Assignment Project Exam Help



Add WeChat powcoder

Solution: $\alpha \rightarrow \text{int}$

type: $\text{int} \rightarrow \text{int}$

$$\frac{\emptyset, x : \alpha, y : \beta \vdash x : \alpha | \emptyset}{\emptyset, x : \alpha \vdash x : \alpha | \emptyset} \quad \text{top}$$
$$\frac{\emptyset, x : \alpha \vdash x : \alpha | \emptyset}{\emptyset, x : \alpha, y : \beta \vdash x : \alpha | \emptyset} \quad \text{middle}$$
$$\frac{\emptyset \vdash (\lambda x. \lambda y. x) : \alpha \Rightarrow (\beta \Rightarrow \alpha) | \emptyset \quad \emptyset \vdash \lambda y. x : \beta | \emptyset}{\emptyset \vdash (\lambda x. \lambda y. x) \lambda y. x : \alpha \Rightarrow \beta | \emptyset} \quad \text{bottom}$$
$$\emptyset \vdash (\lambda x. \lambda y. x) \lambda y. x : \alpha \Rightarrow \beta | \emptyset \quad \emptyset \vdash \text{true} : \text{bool} | \emptyset$$

$\emptyset \vdash (\lambda x. \lambda y. x) \lambda y. x : \alpha \Rightarrow \beta | \emptyset$ true: $\alpha | \alpha = \text{int}; \beta = \text{bool}$

$e_1 \quad e_2$

solution: $\alpha \rightarrow \text{int}$

$\beta \Rightarrow \text{bool}$

type: int

$$\alpha \rightarrow (\alpha \rightarrow \text{int}) = \text{int} \rightarrow \beta$$

- $\text{Unify}(\emptyset) \triangleq []$, the empty substitution
- $\text{Unify}((\tau_1 = \tau_2); E) \triangleq \text{Unify}(E)$ if $\tau_1 = \tau_2$
- $\text{Unify}((\alpha = \tau); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ
- $\text{Unify}((\tau = \alpha); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ
- $\text{Unify}((\tau_1 \rightarrow \tau_2 = \tau'_1 \rightarrow \tau'_2); E) \triangleq \text{Unify}(\tau_1 = \tau'_1; \tau_2 = \tau'_2; E)$
- $\text{Unify}(\tau_1 = \tau_2) \triangleq \text{fail}$ if none of the previous rules apply

$$\text{Unify}(\text{Constraints}) \xrightarrow[\text{(simplify)}]{\text{partial solution}} [\alpha \leftarrow J] \circ (\text{Unify}(\text{Constraints}))$$

1) $\text{Unify}(\alpha = \text{int}; \text{int} = \alpha)$

$$[\alpha \leftarrow \text{int}] \circ \text{Unify}(\text{int} = \alpha; \text{int} = \alpha) \xrightarrow{\uparrow} [\alpha \leftarrow \text{int}] \circ \text{Unify}(\text{int} = \text{int})$$

replaced by int

$$\cdot [\alpha \leftarrow \text{int}] \circ []$$

$\vdash [\alpha \leftarrow \text{int}]$

2) $\text{Unify}(\alpha \rightarrow (\alpha \rightarrow \text{int}) = \text{int} \rightarrow \beta)$

$$\begin{aligned} &= \text{Unify}([\alpha \leftarrow \text{int}; \alpha \rightarrow \text{int}] = \beta) \\ &= [\alpha \leftarrow \text{int}] \circ \text{Unify}(\alpha \rightarrow \text{int} = \beta \quad \boxed{\alpha \leftarrow \text{int}}) \\ &= [\alpha \leftarrow \text{int}] \circ \text{Unify}(\text{int} \rightarrow \text{int} = \beta) \\ &= [\alpha \leftarrow \text{int}] \circ [\beta \leftarrow (\text{int} \rightarrow \text{int})] \circ \text{Unify}(\beta) \\ &= [\alpha \leftarrow \text{int}] \circ [\beta \leftarrow (\text{int} \rightarrow \text{int})] \end{aligned}$$

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

CMPSC 461: Programming Language Concepts

Note 4: Type Inference and Unification

1 Introduction

Type checking is a lightweight technique for proving type safety. However, writing down one type annotation for each variable can be tedious for programmers. *Type inference* is the process of determining the appropriate types for expressions based on how they are used. For example, OCaml knows that in the expression $(x + 3)$, x must be a natural number, because it is an operand of the plus function. A type inference algorithm automatically infers types for variables whose type annotations are missing. For $(x + 3)$, the OCaml type inference engine would assign x with type `int`.

2 Typing Constraints

To understand how type inference works, we first define a simple constraint language that can be used to model a type system. In this constraint language, constraints are equalities on types. The constraint language syntax (written in context-free grammar) is as follows:

$$C ::= C_1; C_2 \mid \tau_1 = \tau_2$$

Assignment Project Exam Help

In this constraint language, a constraint element may be a type variable α whose value is to be solved for, or a type in the simply-typed λ -calculus we defined in Note 3. Constraints C is in general a set of type equality constraints, separated by semicolons ($;$).

<https://powcoder.com>

Validity and Satisfiability A constraint $\tau_1 = \tau_2$ with no type variable is *valid* if types τ_1 and τ_2 are equivalent, defined by the follow induction rules:

Add WeChat powcoder

$$\text{int} = \text{int} \quad \text{bool} = \text{bool} \quad \frac{\tau_1 = \tau'_1 \quad \tau_2 = \tau'_2}{\tau_1 \rightarrow \tau_2 = \tau'_1 \rightarrow \tau'_2}$$

Validity as defined above works for constraints without variables. When constraints mention variables, they are *satisfiable* if there exists a valuation of all variables such that the constraints after value substitution are all valid. For example, a constraint $\alpha = \text{int}$ is satisfiable, since replacing α with `int` gives a valid constraint `int = int`. A constraint $\text{int} = \alpha \rightarrow \alpha$ is unsatisfiable, since no value substitution makes it valid. Constraints $\alpha = \text{int}; \alpha = \text{bool}$ are unsatisfiable, since only one of these two constraints can be satisfied, but not both. We will look at an efficient algorithm that solves constraints later in this lecture.

3 Type Checking as Constraint Solving

A type system with type inference maps naturally into constraint solving, since typing rules are just equality constraints on types. We first extend the simply-typed lambda calculus defined in Note 3 to include variable declaration without type annotation:

$$\text{terms} \qquad e ::= \dots \mid \lambda x . e$$

Here the dots represent all λ -terms defined in Note 3. The added term $\lambda x . e$ defines a function without providing a type for the parameter x .

To formally define type inference, we introduce a new typing relation $\Gamma \vdash e : \tau \mid C$, read as “ e has type τ in type environment Γ if constraints C are satisfiable. In other words, if $\Gamma \vdash e : \tau \mid C$, then expression e has type τ provided that every constraint in C is satisfied.

Here are the new typing rules:

$$\Gamma \vdash n : \text{int} \mid \emptyset \text{ (T-NUM)} \quad \Gamma \vdash \text{true} : \text{bool} \mid \emptyset \text{ (T-TRUE)} \quad \Gamma \vdash \text{false} : \text{bool} \mid \emptyset \text{ (T-FALSE)}$$

$$\frac{\Gamma, x : \tau \vdash x : \tau \mid \emptyset \text{ (T-VAR)} \quad \Gamma \vdash e_1 : \tau \rightarrow \tau_1 \mid C_1 \quad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash e_1 e_2 : \tau_1 \mid C_1; C_2; \tau = \tau_2} \text{ (T-APP)}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau' \mid C \quad \Gamma, x : \alpha_x \vdash e : \tau' \mid C \quad \alpha_x \text{ a fresh variable}}{\Gamma \vdash (\lambda x : \tau . e) : \tau \rightarrow \tau' \mid C} \text{ (T-ABS1)} \quad \frac{}{\Gamma \vdash (\lambda x . e) : \alpha_x \rightarrow \tau' \mid C} \text{ (T-ABS2)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid C_1 \quad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash (e_1 + e_2) : \text{int} \mid C_1; C_2; \tau_1 = \text{int}; \tau_2 = \text{int}} \text{ (T-ADD)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid C_1 \quad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash (e_1 \wedge e_2) : \text{bool} \mid C_1; C_2; \tau_1 = \text{bool}; \tau_2 = \text{bool}} \text{ (T-AND)}$$

Assignment Project Exam Help

- The first four rules are just the same as in Note 3, except that each of them also produces an empty set of constraints.

<https://powcoder.com>

- (T-App) An application expression $e_1 e_2$ represents the result of applying the function represented by e_1 to the argument represented by e_2 . e_1 must be a function of type $\tau \rightarrow \tau_1$ for some τ, τ_1 . Hence, the result type is τ_1 . This rule accumulates constraints C_1, C_2 and produces a new constraint $\tau = \tau_2$, assuming that argument e_2 has type τ_2 . The reason is that formal parameter's type must match real parameter's type.
- (T-Abs1 and T-Abs2) A λ -abstraction is supposed to represent a function. The type of the input should match the annotation in the definition (T-Abs1). When no annotation is provided, a fresh type variable α_x is generated for the input. The type τ' of the result is the type of the body. This rule does not generate new constraints; it just carries the constraints generated by the function body.
- The last two rules say the types of operands of add (resp. and) must be equivalent to type int (resp. bool), and the result has type int (resp. bool).

Using these typing rules, the proof tree of every term generates a set of constraints to be solved. For example, consider the term $\lambda x . (x + 1)$, which takes x , whose type is to be inferred, and returns $x + 1$. Here is the proof tree of this term:

$$\frac{\frac{x : \alpha_x \vdash x : \alpha_x \mid \emptyset \quad x : \alpha_x \vdash 1 : \text{int} \mid \emptyset}{x : \alpha_x \vdash (x + 1) : \text{int} \mid \alpha_x = \text{int}; \text{int} = \text{int}}}{\vdash (\lambda x . (x + 1)) : \alpha_x \rightarrow \text{int} \mid \alpha_x = \text{int}; \text{int} = \text{int}}$$

It is easy to check that the top-level constraints ($\alpha_x = \text{int}; \text{int} = \text{int}$) are satisfiable, with solution $\alpha_x \leftarrow \text{int}$. Hence, the term $\lambda x . (x + 1)$ has type $\text{int} \rightarrow \text{int}$, and the inferred type of x is int .

Consider another term $(\lambda x . \lambda y . x) 2 \text{ true}$, which evaluates to 2. Here is the proof tree of this term:

$$\begin{array}{c}
 \frac{x : \alpha_x, y : \alpha_y \vdash x : \alpha_x \mid \emptyset}{x : \alpha_x \vdash (\lambda y . x) : \alpha_y \rightarrow \alpha_x \mid \emptyset} \\
 \frac{}{\vdash (\lambda x . \lambda y . x) : \alpha_x \rightarrow \alpha_y \rightarrow \alpha_x \mid \emptyset} \quad \vdash 2 : \text{int} \mid \emptyset \\
 \frac{}{\vdash ((\lambda x . \lambda y . x) 2) : \alpha_y \rightarrow \alpha_x \mid \alpha_x = \text{int}} \quad \vdash \text{true} : \text{bool} \mid \emptyset \\
 \hline
 \vdash (((\lambda x . \lambda y . x) 2) \text{ true}) : \alpha_x \mid \alpha_x = \text{int}; \alpha_y = \text{bool}
 \end{array}$$

It is easy to check that the top-level constraints $\alpha_x = \text{int}; \alpha_y = \text{bool}$ are satisfiable. Hence, the term $((\lambda x . \lambda y . x) 2) \text{ true}$ has type `int`, and the inferred type of x is `int`; the inferred type of y is `bool`.

4 Unification

Constraints defined in this lecture can be solved by a very general mechanism called unification. Briefly, unification is the process of finding a substitution that makes two given terms equal. A *type substitution* is a finite map from type variables to types. For example, we write $[\alpha \leftarrow \text{int}, \beta \leftarrow (\text{int} \rightarrow \text{int})]$ for the substitution that maps type variable α to `int`, and type variable β to $\text{int} \rightarrow \text{int}$.

The essential task of unification is to find a substitution S that unifies two given terms (that is, makes them equal). Let us write $[C]S$ for the result of applying the substitution S to the terms. For example, the substitution $[\alpha \leftarrow \text{int}, \beta \leftarrow (\text{int} \rightarrow \text{int})]$ unifies terms $(\alpha \rightarrow (\beta \rightarrow \text{int}))$ and $(\text{int} \rightarrow \beta)$:

$$\begin{aligned}
 & (\alpha \rightarrow (\alpha \rightarrow \text{int})) = \text{int} \rightarrow \beta [\alpha \leftarrow \text{int}, \beta \leftarrow (\text{int} \rightarrow \text{int})] \\
 & = (\text{int} \rightarrow (\text{int} \rightarrow \text{int})) = \text{int} \rightarrow (\text{int} \rightarrow \text{int})
 \end{aligned}$$

In general, the same variable may occur in both the domain and range of a substitution. In that case, the intention is that the substitutions are performed simultaneously. For example the substitution $[\alpha \leftarrow \text{int}, \beta \leftarrow (\text{int} \rightarrow \alpha)]$ maps β to $\text{int} \rightarrow \alpha$ instead of $\text{int} \rightarrow \text{int}$. Unlike the substitution in β -reduction, we do not need to worry about avoiding variable capture, since there are no constructs in the language that bind type variables.

Given two substitutions S_1 and S_2 , we write $S_1 \circ S_2$ for their composition. For example, $([\alpha \leftarrow \text{int}] \circ [\beta \leftarrow \text{bool}]) = [\alpha \leftarrow \text{int}, \beta \leftarrow \text{bool}]$. Note that composition performs subststation in sequence. For example, $([\alpha \leftarrow \beta] \circ [\beta \leftarrow \gamma] \circ [\gamma \leftarrow \text{bool}]) = [\alpha \leftarrow \text{bool}, \beta \leftarrow \text{bool}, \gamma \leftarrow \text{bool}]$.

To solve a set of constraints C , we need to find a substitution that unifies C . More specifically, suppose that $\Gamma \vdash e : \tau \mid C$; a solution for constraints C is a substitution S such that all equalities in $C[S]$ are valid. If there is no substitution that satisfies C , then we know that e is not typeable.

4.1 Unification Algorithm

The unification algorithm is known as Robinson's algorithm. The unification algorithm is given in terms of a function `Unify` that takes a set of constraints C and produces a substitution, if it exists. If C is a set of constraints, then $C[\alpha \leftarrow t]$ denotes the result of applying the substitution $\alpha \leftarrow t$ to all the constraints in C . The algorithm is defined as follows:

- $\text{Unify}(\emptyset) \triangleq []$, the empty substitution
- $\text{Unify}((\tau_1 = \tau_2); E) \triangleq \text{Unify}(E)$ if $\tau_1 = \tau_2$
- $\text{Unify}((\alpha = \tau); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ

- $\text{Unify}((\tau = \alpha); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ
- $\text{Unify}((\tau_1 \rightarrow \tau_2 = \tau'_1 \rightarrow \tau'_2); E) \triangleq \text{Unify}(\tau_1 = \tau'_1; \tau_2 = \tau'_2; E)$
- $\text{Unify}(\tau_1 = \tau_2) \triangleq \text{fail}$ if none of the previous rules apply

The check that α is not a free variable of the other type ensures that the algorithm does not produce a cyclic substitution (e.g., $[\alpha \leftarrow (\alpha \rightarrow \alpha)]$), which does not make sense since our language does not support recursive types.

For example, $\text{Unify}(\alpha_x = \text{int}; \alpha_y = \text{int}) = [\alpha_x \leftarrow \text{int}] \circ \text{Unify}(\alpha_y = \text{int}) = [\alpha_x \leftarrow \text{int}] \circ [\alpha_y \leftarrow \text{int}] = [\alpha_x \leftarrow \text{int}, \alpha_y \leftarrow \text{int}]$. On the other hand, $\text{Unify}(\alpha_x = \text{int}; \alpha_x = \text{bool}) = [\alpha_x \leftarrow \text{int}] \circ \text{Unify}(\text{int} = \text{bool}) = \text{fail}$.

The unification algorithm always terminates. Moreover, it produces a solution if and only if a solution exists. The solution found is the most general solution, in the sense that if $S = \text{Unify}(C)$ and S' is a solution to C , then there is some S'' such that $S' = S'' \circ S$.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

31

Assignment Project Exam Help

Types <https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Midterm 2

Midterm 2 is scheduled to Nov. 13th 6:15 to 7:30
Assignment Project Exam Help

Conflict exam? Send request to TA Zeyu (dxd437@psu.edu)
and me (dbz5017@psu.edu) by Nov. 11th
Add WeChat powcoder

HW6 due next Wednesday midnight, ***no late submission***

Midterm 2

Midterm 2 covers materials from HW3 to HW6, which is the same as lecture 17 to lecture 31 (this lecture)

Assignment Project Exam Help

Next Wednesday: <https://powcoder.com> Midterm 2

Add WeChat powcoder

Practice questions will be posted soon

Changes to the ZOOM exam

Midterm 2 will be like Midterm 1, with a few changes

- 50 minutes exam → 60 minutes exam
- 10pt bonus question → 10pt extra question (110 points in total, graded as 100 points exam)
- Students have 15 minutes to copy questions (**without working on them!**) before the exam starts
- Assign students to different ZOOM rooms, rather than using breakout rooms

A detailed instruction will be posted soon.

Function performs an operation for different values
of the same type

Assignment Project Exam Help

```
class IntArrayList {  
    ...  
    boolean add (int i);  
    int remove(int idx);  
}  
  
class DoubleArrayList {  
    ...  
    boolean add (double d);  
    double remove(int idx);  
}
```

(Polymorphic) Functions takes values of different types

Java used to do this (before Java 1.5):

```
Assignment Project Exam Help
class ArrayList {
    ...
    boolean add (Object o);
    Object remove (int idx);
}
... https://powcoder.com
Add WeChat powcoder
```

Does it work?

Assignment Project Exam Help

```
List langList = new ArrayList();
langList.add(new Language("Java"));
Language java = (Language) langList.remove(0);
langList.add(new AddWeChat powcoder()); // Whoops !
Language c = (Language) langList.remove(0); // Error
```

Compiler doesn't know langList should
only contain values of type Language

Generics

A facility added to Java 5.0, which allows “a type or method to operate on objects of various types while providing compile-time type safety.”

<https://powcoder.com>

Not essentially an object-oriented idea
Add WeChat powcoder
Not originated in Java

Polymorphism: function with multiple forms

Parametric polymorphism (fun. with a set of types)

- Explicit para. polymorphism (Generics, aka. Assignment Project Exam Help Templates in C++)

Java

```
class ArrayList<T>
{
    ...
    boolean add (T o);
    T remove (int idx);
}
```

<https://powcoder.com>

```
C++
template <class T>
T GetMax (T a, T b) {
    T result;
    result = (a>b) ? a : b;
    return (result);
}
```

Polymorphism

Parametric polymorphism (fun. with a set of types)

- Explicit para. polymorphism
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Implicit para. polymorphism (Lisp, Haskell, ML)
<https://powcoder.com>

```
(define (min a b) (if (< a b) a b))
```

Add WeChat powcoder

No mention
of type

In Haskell

```
min a b = if a < b then a else b
```

Type Parameterization

```
ArrayList<Integer> intLst = new ArrayList<Integer>();  
ArrayList<String> strLst = new ArrayList<String>();
```

Assignment Project Exam Help

ArrayList<Integer> <https://powcoder.com>

- an application of a type-level function (ArrayList)
- to the type parameter Integer
- gives an array list of integers

Add WeChat powcoder

The idea of generics is to allow user-defined parameterized types

User-Defined Parameterized Type

Java

```
class ArrayList<T> {  
    ...  
    boolean add (T o);  
    T remove (int idx);  
}
```

Formal type
parameter

C++

```
template<class T>  
class ArrayList {  
    ...  
    bool add (T o);  
    T remove (int idx);  
}
```

Formal type
parameter

Intuitively, we defined a type-level function

Generic Functions

Java

Formal type
parameter

```
T max<T> (T a, T b) {  
    T result;  
    result = (a.compareTo(b)  
              >=0) ? a : b;  
    return (result);  
}
```

C++

Formal type
parameter

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
template <class T>  
T GetMax (T a, T b) {  
    T result;  
    result = (a>b) ? a : b;  
    return (result);  
}
```

Intuitively, a function that takes a type, and parameters

Generics Enforce Type-Safety

Assignment Project Exam Help

```
List<Language> langList = new ArrayList<Language>();  
langList.add(new Language("Java")); // OK  
Language java = (Language) langList.remove(0);  
langList.add(new System("Linux")); // Type error!  
Language c = (Language) langList.remove(0);
```

Compiler knows langList only contain values
of type Language (no run-time errors)

Implementing Generics

Static mechanism (Ada, C++)

- Compiler generates separate copy for every unique instance
- Each copy is type-checked separately

<https://powcoder.com>

```
template <class T>
T GetMax (T a, T b) {
    T result;
    result = (a>b) ? a : b;
    return (result);
}
int i = GetMax(1,2);
char c = GetMax('c','e');
```

Defined by programmer

```
int GetMax (int a,int b) {
    int result;
}
char GetMax (char a, char b) {
    char result;
...
}
int i = GetMax(1,2);
char c = GetMax('c','e');
```

Generated/Checked by the compiler

Implementing Generics

Dynamic mechanism (aka. type erasure in Java)

- Type system checks type safety
- All instances share one code, without generics!

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Type Erasure in Java

```
class ArrayList<T> {  
    ...  
    boolean add (T o) { ... } ;  
    T remove (int idx) { ... } ;  
}
```

Assignment Project Exam Help

```
ArrayList<Language> langLst = ...;  
langLst.add(new Language("Java"));  
Language java=langLst.remove(0);
```

Defined by programmer

Add WeChat powcoder

```
class ArrayList {  
    ...  
    boolean add (Object o) { ... } ;  
    Object remove (int idx) { ... } ;  
}  
ArrayList langLst = ...;  
langLst.add(new Language("Java"));  
Language java=  
(Language) langLst.remove(0);
```

Executed by JVM

Type system
ensures all dynamic
casts are safe

Pros and Cons of Type Erasure

Pros

- Backward compatibility
- One code copy shared by all instances

Cons

- Cannot use type parameters around <https://powcoder.com>

```
class Example<T> {  
    void method(Object item) {  
        if (item instanceof T) { ... } // cannot compare to T  
        T anotherItem = new T(); // cannot use constructor  
        T[] itemArray = new T[10]; // cannot create T[10]  
    } }
```

32

Assignment Project Exam Help
Program Verification
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Correctness

A program implements the desired property for all possible inputs

Assignment Project Exam Help

E.g.,

<https://powcoder.com>

Functional correctness: a program calculates $n!$

Add WeChat powcoder

Type safety: no typing errors at run time

Memory safety: no buffer overflow in a program

Security: no information leakage, no violation of integrity, ...

We will focus on **functional correctness** in this course

Functional Correctness

```
r:=0, i:=0;  
while (i<n) {  
    r := r+2;  
    i++;  
}
```

Assignment Project Exam Help

<https://powcoder.com>

This code ensures $r = 2 \times n$ when $n \geq 0$?

- **Testing:** assert $r := 2 \times n$ and execute with different values of n (cannot cover all inputs in general)
- **Verification:** prove $r = 2 \times n$ for any possible n

Program and Logics

Is a program correct (e.g., is the result $n!$)?

Assignment Project Exam Help

We need to formally specify

<https://powcoder.com>

- 1) The desired property
- 2) The behavior of program

Add WeChat powcoder

Logics as our specification language

Background: Propositional Logic

Proposition: statements that can either be true or false

E.g., $1 > 0$ (true), $0 \times 5 = 0$ (true), $5 - 1 = 5$ (false)

Assignment Project Exam Help

Composed propositions: with prop. p and q ,

<https://powcoder.com>

And: $p \wedge q$ true iff both p and q are true

Or: $p \vee q$ false iff both p and q are false

not: $\neg p$ true iff p is false

E.g., $1 > 0 \wedge 0 \times 5 = 0$ (true)

$0 \times 5 = 0 \vee 5 - 1 = 5$ (true)

Implication

A proposition p implies another proposition q (written as $p \Rightarrow q$) iff ($\neg p \vee q$)
(intuitively, q is true whenever p is true)

E.g., $7 > 5 \Rightarrow 5 > 3$ (true)

$1 > 2 \Rightarrow 2 > 3$ (true, due to the false condition)

$2 > 1 \Rightarrow 2 > 3$ (false)

Derivation:

Modus Ponens: given $p \Rightarrow q$ and p , we have q

(First-Order) Predicate Logic

A formula can mention bounded variables

For all (Universally quantified): $\forall x.p$

E.g., $\forall x. x = 5$ (false)
<https://powcoder.com>

Exists (Existentially quantified): $\exists x.p$

E.g., $\exists x. x = 5$ (true)

(We assume the domain of integers for simplicity)

(First-Order) Predicate Logic

E.g., $\forall x. (x > 5 \Rightarrow x > 3)$ (true)

$\forall x. (x > 1 \Rightarrow x > 3)$ (false, consider $x = 2$)

$\exists x. (x > 1 \Rightarrow x > 3)$ (true, consider $x = 4$)

<https://powcoder.com>

Truth value of a formula can be *proved* based on derivation rules from predicate logic and number theory

We only use simple formulas in this course; in general, some tools (e.g., an SMT solver) can automatically tell the truth value of many formulas

Program and Logics

We need to formally specify

- 1) The desired property

Assignment Project Exam Help

```
int Max(int a, int b) {  
    int m;  
    if (a > b) m = a;  
    else m := b;  
    return m;  
}
```

Assignment

Precondition (true) function symbol in logics

Postcondition ($m = \max(a, b)$)

Program and Logics

We need to formally specify

- 1) The desired property

Assignment Project Exam Help

```
int factorial(int n) {  
    int r=1, i=n;  
    while (i>0) {  
        r := r*i;  
        i --;  
    }  
    return r;  
}
```

Precondition ($n \geq 0$)

Postcondition ($r = n!$)

Program and Logics

Is a program correct?

Assignment Project Exam Help

We need to formally specify

<https://powcoder.com>

- 1) The desired property
- 2) The behavior of program

Add WeChat powcoder

Informally ...

```
x := 5;  
y := 1;
```

Assignment Project Exam Help

After second assignment, we know $\{x = 5, y = 1\}$
<https://powcoder.com>

Why?

Add WeChat powcoder

1. Initially, we assume nothing
2. After the first assignment, we know $\{x = 5\}$
3. After the second assignment, we know $\{y = 1\}$ is true as well

Formalizing the Reasoning

```
x := 5;  
y := 1;
```

1. Initially, we assume nothing
2. After the first assignment, we know $\{x = 5\}$
3. After the second assignment, we know $\{y = 1\}$ is true as well

The reasoning:

$$\{\text{true}\}_{x:=5} \quad \{x = 5\} \quad y := 1 \quad \{x = 5 \wedge y = 1\}$$

Each predicate specifies the assertion that must be true before/after a statement

Hoare Triple

Assertion: a predicate that describes the **state** of a program at a point in its execution

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Hoare Triple: $\{P\} s \{Q\}$

Precondition P Add Assertion before execution

Postcondition Q : an assertion after execution

Program s : program being analyzed

A triple is **valid** If we start from a state satisfying P , and execute s , then final state must satisfy Q

Examples

{true} $x := 5 \{x = 5\}$ valid

{ $y = 6$ } $x := 5 \{x = 5, y = 6\}$ valid

{true} $x := 5 \{x < 10\}$ valid

{ $x = y$ } $x := x + 3 \{x = y + 3\}$ valid

{ $x = a$ } if ($x < 0$) then $x := -x \{x = |a|\}$ valid

All of these triples are valid

{true} $x := 5 \{x = 0\}$ invalid

{true} $x := 5 \{x < 0\}$ invalid

{true} $x := x + 1 \{x > 0\}$ invalid

Program Correctness

A program is correct (w.r.t. pre/postcondition) if the corresponding Hoare triple is valid

Assignment Project Exam Help

```
{true}
int m;
if (a>b)    m:=a;
else          m:=b;
{m=max(a,b)}
```

<https://powcoder.com>

Add WeChat powcoder

```
{n >= 0}
int r:=1, i:=n;
while (i>0) {
    r := r*i;
    i--;
}
{r = n!}
```

How can we tell the validity of a Hoare triple?

CMPSC 461: Programming Language Concepts

Midterm 2 Practice Questions

Use these problems in addition to Assignment 4, 5 and 6 to prepare for the 2nd midterm.

Problem 1 For each of the following Scheme programs, circle all x's that refer to (i.e., are in the scope of) the definition of x at the FIRST LINE. You don't need to circle anything if no such x exists.

```
(let ((x 1))
  (let ((x 2))
    (+ x y)))
```

```
(let ((x 3))
  (let ((x 4) (y x))
    (+ x y)))
```

```
(let ((x 5) (y 6))
  (let* ((y x) (x y))
    (+ x y)))
```

Problem 2 What is the difference between static, stack, and heap allocation and how do they affect the lifetime of a variable?

Problem 3 What is a tail-recursive function? What makes it an interesting concept?

Problem 4 What outputs are produced by the following pseudo code if the language uses static scoping? What are the outputs if the language uses dynamic scoping?

```
a=2; b=3;
int f1(a) {
  return a + b;
}
int f2(b) {
  return 2 * f1(b);
}
print f1(a) * f2(a);
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Problem 5 Consider the following pseudo code.

```
int a=0;
void A(int m) {
  print a;
  m = a;
}
void main () {
  int a=1, b=2;
  A(b);
  print b;
}
```

1. What are the outputs if the language uses static scoping and all parameters are passed by value?
2. What are the outputs if the language uses dynamic scoping and all parameters are passed by reference?

Problem 6 Use the following typing rules to write down the proof tree for the term $((\lambda x : \text{bool} . (\lambda y : \text{bool} . x \wedge y)) \text{ true})$.

Typing rules: $\Gamma \vdash \text{true} : \text{bool}$ (T-TRUE) $\Gamma \vdash \text{false} : \text{bool}$ (T-FALSE) $\Gamma, x : \tau \vdash x : \tau$ (T-VAR)

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} \text{ (T-APP)} \quad \frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash (\lambda x : \tau . e) : \tau \rightarrow \tau'} \text{ (T-ABS)}$$

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash (e_1 \wedge e_2) : \text{bool}} \text{ (T-AND)}$$

Problem 7 Follow the constraint unification rules in Lecture Note 4 to solve the following constraint

$$(\text{int} \rightarrow \alpha = \beta \rightarrow \beta; \beta = \text{int})$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Review

Problem 6 Use the following typing rules to write down the proof tree for the term $((\lambda x : \text{bool} . (\lambda y : \text{bool} . x \wedge y)) \text{ true}$.

Typing rules: $\Gamma \vdash \text{true} : \text{bool}$ (T-TRUE) $\Gamma \vdash \text{false} : \text{bool}$ (T-FALSE) $\Gamma, x : \tau \vdash x : \tau$ (T-VAR)

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} \text{ (T-APP)} \quad \frac{\Gamma \vdash e : \tau' \quad \Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash (\lambda x : \tau . e) : \tau \rightarrow \tau'} \text{ (T-ABS)}$$

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash (e_1 \wedge e_2) : \text{bool}} \text{ (T-AND)}$$

① build a proof tree based CFG

② fill in missing types

$$\begin{array}{c} J \rightarrow J' \\ \text{bool} \quad \text{bool} \rightarrow \text{bool} \end{array}$$

Assignment Project Exam Help

$\emptyset, x : \text{bool}, y : \text{bool} \vdash x : \boxed{\text{bool}}$ $\emptyset, x : \text{bool}, y : \text{bool} \vdash y : \boxed{\text{bool}}$

$\emptyset, x : \text{bool}, y : \text{bool} \vdash x \wedge y : \boxed{\text{bool}} \leftarrow T$

$\emptyset, x : \text{bool} \vdash (\lambda y : \text{bool}. (x \wedge y)) : \boxed{\text{bool} \rightarrow \text{bool}}$

$\emptyset \vdash (\lambda x : \text{bool}. (\lambda y : \text{bool}. (x \wedge y))) : \boxed{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})}$ $\emptyset \vdash \text{true} : \boxed{\text{bool}}$

$\emptyset \vdash ((\lambda x : \text{bool}. (\lambda y : \text{bool}. (x \wedge y))) \text{ true}) : \boxed{\text{bool} \rightarrow \text{bool}}$

Add WeChat powcoder

Problem 7 Follow the constraint unification rules in Lecture Note 4 to solve the following constraint

$$(\text{int} \rightarrow \alpha = \beta \rightarrow \beta; \beta = \text{int})$$

- $\text{Unify}(\emptyset) \triangleq []$, the empty substitution
 - $\text{Unify}((\tau_1 = \tau_2); E) \triangleq \text{Unify}(E)$ if $\tau_1 = \tau_2$
- ④
- $\text{Unify}((\alpha = \tau); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ
 - ②
 - $\text{Unify}((\tau = \alpha); E) \triangleq [\alpha \leftarrow \tau] \circ (\text{Unify}(E[\alpha \leftarrow \tau]))$ if α is not a free variable in τ
 - ①
 - $\text{Unify}((\tau_1 \rightarrow \tau_2 = \tau'_1 \rightarrow \tau'_2); E) \triangleq \text{Unify}(\tau_1 = \tau'_1; \tau_2 = \tau'_2; E)$
 - $\text{Unify}(\tau_1 = \tau_2) \triangleq \text{fail}$ if none of the previous rules apply

Assignment Project Exam Help

$$\text{Unify } (\text{int} \rightarrow \alpha = \beta \rightarrow \beta; \beta = \text{int})$$

$$= \text{Unify}(\text{int} \rightarrow \alpha = \beta \rightarrow \beta; \beta = \text{int})$$

$$= [\beta \leftarrow \text{int}] \circ \text{Unify}(\alpha = \beta; \beta = \text{int})$$

$$= [\beta \leftarrow \text{int}] \circ \text{Unify}(\alpha = \text{int}, \text{int} = \text{int})$$

$$= [\beta \leftarrow \text{int}] \circ [\beta \leftarrow \text{int}] \circ \text{Unify}(\text{int} = \text{int})$$

$$= [\beta \leftarrow \text{int}] \circ [\alpha \leftarrow \text{int}] \circ \text{Unify}(\text{int} = \text{int})$$

$$= [\beta \leftarrow \text{int}] \circ [\alpha \leftarrow \text{int}] \circ \text{Unify}(\emptyset)$$

$$= [\beta \leftarrow \text{int}] \circ [\alpha \leftarrow \text{int}]$$

<https://powcoder.com>

Add WeChat powcoder

Problem 4 What outputs are produced by the following pseudo code if the language uses static scoping? What are the outputs if the language uses dynamic scoping?

```
a=2; b=3;
int f1(a) {
    return a + b;
}
int f2(b) {
    return 2 * f1(b);
}
print f1(a) * f2(a);
```

key: to draw symbol table trees

Tables:

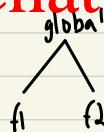
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Static Scoping:

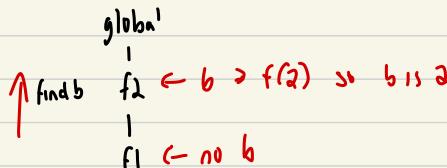
```
call f1(2)
return 2+3
call f2(2)
return 2×f1(2)
                (2+3)
print 5×10 = 50
```



Dynamic Scoping:

```
call f1(2)
return 2+3
2×f1(2)
                (2+2×4)
```

print 5×(2×4) = 40



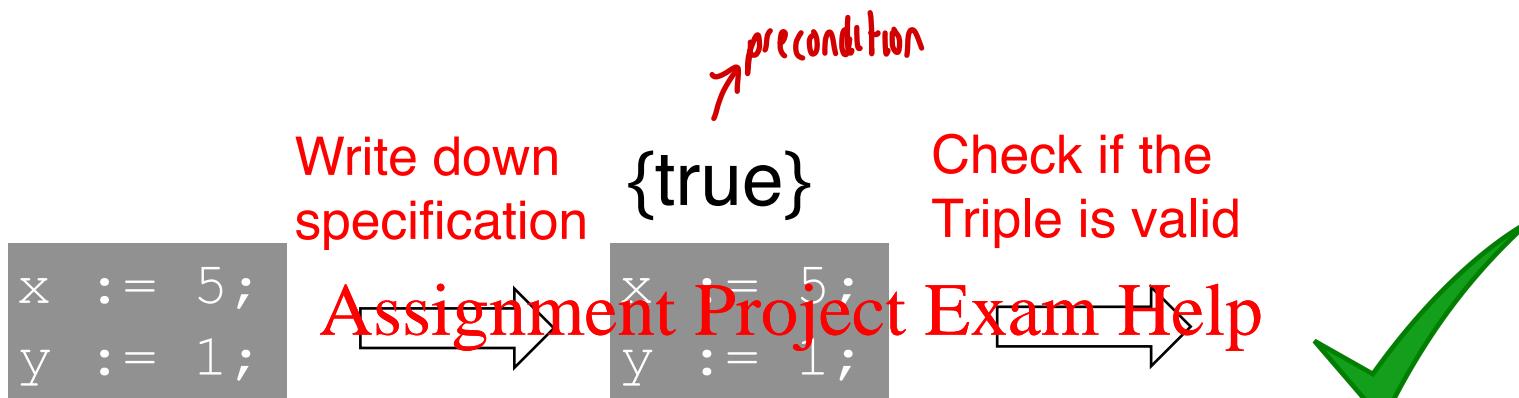
#33

Assignment Project Exam Help
Program Verification
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Overview of Program Verification



$x+y>5$ after execution from any initial state?

Add ~~WeChat known as~~ ^{post condition} **Hoare triple** Program is verified if the triple is valid

A triple $\{P\} s \{Q\}$ is **valid** If we start from a state satisfying P , and execute s , then final state must satisfy Q

Goal: check if $\{P\} s \{Q\}$ is valid

$\{\text{true}\} x := 5 \{x = 5\}$ ✓ *most useful*

$\{\text{true}\} x := 5 \{x = 5 \vee x = 2\}$ ✓

$\{\text{true}\} x := 5 \{x > 0\}$ ✓ *Assignment Project Exam Help*

$\{\text{true}\} x := 5 \{x < 10\}$ ✓ *https://powcoder.com*

Observation: some postconditions are more useful
Add WeChat powcoder

$x = 5 \Rightarrow x = 5 \vee x = 2$

$x = 5 \Rightarrow x > 0$

$x = 5 \Rightarrow x < 10$

Need to compute the ***strongest*** postcondition

Compute strongest postcondition and check the truth value

$$sp(x := 5; y := 1, true)$$

Assignment Project Exam Help $x + y > 5$?

Write down specification <https://powcoder.com> {true}

```
x := 5;  
y := 1;
```



$$\{x+y>5\}$$

$x+y>5$ after execution from any initial state?

Program is verified if the triple is valid

Goal: check if $\{P\} s \{Q\}$ is valid
Method 1: check $\text{sp}(s, P) \Rightarrow Q$

Strongest Postcondition

$\text{sp}(s, P)$ is the **strongest postcondition** of s , w.r.t. P
Property: $\{P\} s \{Q\}$ is valid iff $\text{sp}(s, P) \Rightarrow Q$

Add WeChat powcoder

Hence, validity of a triple $\{P\} s \{Q\}$ is equivalent to
the truth value of proposition $\text{sp}(s, P) \Rightarrow Q$

Assignment Rule (Floyd's Axiom)

$$\text{sp}(x := e, P) = (\exists v. \underset{\substack{\text{value of } x \text{ before assignment} \\ \text{new value of } x}}{(x = (e[x \leftarrow v]))} \wedge (P[x \leftarrow v]))$$

Assignment Project Exam Help

Substitute x with v in e

Examples: <https://powcoder.com>

$$\text{sp}(x := 5, \text{true}) = (\exists v. (x = 5) \wedge \text{true}) = (x = 5)$$

$$\begin{aligned} \text{sp}(x := x + 3, x = y) &= (\exists v. (x = v + 3) \wedge (v = y)) \\ &= (x = y + 3) \end{aligned}$$

$$\begin{array}{ll} x+3[x \leftarrow v] & x=y[x \leftarrow v] \\ : v+3 & v=y \end{array}$$

Composition Rule

$\{ \rho \}$
 s_1
 $\leftarrow s\rho(s_1, \rho)$
 s_2
 $-s\rho(s_2, s\rho(s_1, \rho))$

$$sp(s_1; s_2, P) = sp(s_2, sp(s_1, P))$$

Assignment Project Exam Help

x := 5;
y := 1;

<https://powcoder.com>

$$\begin{aligned} sp(x := 5; y := 1, \text{true}) &= sp(y := 1, sp(x := 5, \text{true})) \\ &= sp(y := 1, x = 5) \text{ (previous slide)} \\ &= (\exists v. (y = 1) \wedge (x = 5)) \\ &= (y = 1) \wedge (x = 5) \end{aligned}$$

Composition Rule

$$\text{sp}(s_1; s_2, P) = \text{sp}(s_2, \text{sp}(s_1, P))$$

Assignment Project Exam Help

x := 5;
x := 2;

<https://powcoder.com>

$$\begin{aligned} \text{sp}(x := 5; x := 2, \text{true}) &= \text{sp}(x := 2, \text{sp}(x := 5, \text{true})) \\ &= \text{sp}(x := 2, x = 5) \\ &= (\exists v. (x = (2[x \leftarrow v]))) \wedge (x = 5[x \leftarrow v])) \\ &= (\exists v. (x = 2) \wedge (v = 5)) \\ &= (x = 2) \end{aligned}$$

Add WeChat  powcoder

The existential quantifier complicates the formula ...

```

{true}
x := 5;
y := 1;
{ (y=1) ∧ (x=5) }

```

$$\begin{aligned}
\text{sp}(x := 5 ; y := 1, \text{true}) &= \text{sp}(y := 1, \text{sp}(x := 5, \text{true})) \\
&= \text{sp}(y := 1, x = 5) \\
&= (\exists v. (y = 1) \wedge (x = 5)) \\
&= (y \neq 1) \wedge (x = 5) ?
\end{aligned}$$

Add WeChat powcoder

The existential quantifier complicates the formula ...

Goal: check if $\{P\} s \{Q\}$ is valid
 Method 1: check $\text{sp}(s, P) \Rightarrow Q$

```

{true}
x := 5;
y := 1;
{ (y=1) ∧ (x=5) }

```

$$\begin{aligned}
\text{sp}(x := 5 ; y := 1, \text{true}) &= \text{sp}(y := 1, \text{sp}(x := 5, \text{true})) \\
&= \text{sp}(y := 1, x = 5) \\
&= (y = 1) \wedge (x = 5)
\end{aligned}$$

Assignment Project Exam Help

The reasoning:

https://powcoder.com
← {true} x := 5 {x = 5} y := 1 {x = 5 ∧ y = 1}
Add WeChat powcoder →

sp computation (forward):

Backward?



Goal: check if $\{P\} s \{Q\}$ is valid

Method 1: check $\text{sp}(s, P) \Rightarrow Q$

Goal: check if $\{P\} s \{Q\}$ is valid

Method 1: check $\text{sp}(s, P) \Rightarrow Q$

Method 2: check $P \Rightarrow \text{wp}(s, Q)$



Weakest Precondition

$\text{wp}(s, Q)$ is the weakest precondition of s , w.r.t. Q

Property: $\{P\} s \{Q\}$ is valid iff $P \Rightarrow \text{wp}(s, Q)$

Add WeChat powcoder

Hence, validity of a triple $\{P\} s \{Q\}$ is equivalent to
the truth value of proposition $P \Rightarrow \text{wp}(s, Q)$

Assignment Rule (Hoare's Axiom)

$$\text{wp}(x := e, Q) = Q[x \leftarrow e]$$

Assignment Project Exam Help

Examples: <https://powcoder.com>

$$\text{wp}(x := 5, x = 5) = (5 = 5) = (\text{true})$$

$$\begin{aligned} \text{wp}(x := x + 3, x = y + 3) &= (x + 3 = y + 3) \\ &= (x = y) \end{aligned}$$

$$x = y + 3 [x \leftarrow x + 3]$$

$$x + 3 = y + 3$$

This rule is simpler than Floyd's axiom, hence weakest precondition is used in most systems

Composition Rule

$s_1(s_1, wp(s_2, Q))$
 $wp(s_2, Q)$
 s_2
 $\{Q\}$

$$wp(s_1 ; s_2, Q) = wp(s_1, wp(s_2, Q))$$

{true}

x := 5;

y := 1;

{ (y=1) \wedge (x=5) }

Assignment Project Exam Help

<https://powcoder.com>

$$\begin{aligned} & wp(x := 5 ; y := 1, (x = 5) \wedge (y = 1)) \\ &= wp(x := 5, wp(y := 1, (x = 5) \wedge (y = 1))) \\ &= wp(x := 5, (x = 5) \wedge (1 = 1)) \\ &= (5 = 5) \wedge (1 = 1) \\ &= \text{true} \end{aligned}$$

Add WeChat $powcoder$

$Q[x \leftarrow s]$

$Q[y \leftarrow 1]$

Composition Rule

$$\text{wp}(s1 ; s2, Q) = \text{wp}(s1, \text{wp}(s2, Q))$$

```
{true}  
x := 5;  
x := 2;  
{x=2}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

$$\begin{aligned} & \text{wp}(x := 5 ; x := 2, x = 2) \\ &= \text{wp}(x := 5, \text{wp}(x := 2, x = 2)) \\ &= \text{wp}(x := 5, 2 = 2) \\ &= (2 = 2) \\ &= \text{true} \end{aligned}$$

$\text{Q}[x \leftarrow 2]$

$\text{Q}[x \leftarrow 5]$

Branch Rule

$$\begin{array}{l} E \Rightarrow S_1 \{ Q \} \\ \neg E \Rightarrow S_2 \{ Q \} \end{array}$$

$\text{wp}(\text{if}(E) \ s1 \ \text{else} \ s2, Q) =$
 $(E \Rightarrow \text{wp}(s1, Q) \wedge \neg E \Rightarrow \text{wp}(s2, Q))$

```
{ true }
if  (x>0)
    y := x;
else
    y := -x;
{ y≥0 }
```

<https://powcoder.com>
 $\text{wp}(P, y \geq 0)$
 $\models x > 0 \Rightarrow \text{wp}(y := x, y \geq 0) \wedge$
 $\neg(x > 0) \Rightarrow \text{wp}(y := -x, y \geq 0))$
 $= (x > 0 \Rightarrow x \geq 0) \wedge (x \leq 0 \Rightarrow -x \geq 0)$
 $= \text{true}$

Computing WP

$$\text{wp}(x := e, Q) = Q[x \leftarrow e]$$

$$\text{wp}(s_1 ; s_2, Q) = \text{wp}(s_1, \text{wp}(s_2, Q))$$

$$\text{wp}(\text{if}(E) s_1 \text{else } s_2, Q) = \\ (E \Rightarrow \text{wp}(s_1, Q) \wedge \neg E \Rightarrow \text{wp}(s_2, Q))$$

$$\text{wp}(\text{nop}, Q) = Q$$

#34

Assignment Project Exam Help
Program Verification
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Goal: check if $\{P\} s \{Q\}$ is valid

Method 1: check $\text{sp}(s, P) \Rightarrow Q$

Method 2: check $P \Rightarrow \text{wp}(s, Q)$

$\text{wp}(x := e, Q) = Q[x \leftarrow e]$

$\text{wp}(s_1 ; s_2, Q) = \text{wp}(s_1, \text{wp}(s_2, Q))$

$\text{wp}(\text{if}(E) s_1 \text{else } s_2, Q) =$

$(E \Rightarrow \text{wp}(s_1, Q) \wedge \neg E \Rightarrow \text{wp}(s_2, Q))$

$\text{wp}(\text{nop}, Q) = Q$

A dummy operation
that has no effects

```

{ x>0 }
x := x+1;
y := x * (x+5);
{ y>0 }

```

$$\begin{aligned}
\text{wp}(x := e, Q) &= Q[x \leftarrow e] \\
\text{wp}(s_1 ; s_2, Q) &= \text{wp}(s_1, \text{wp}(s_2, Q)) \\
\text{wp}(\text{if}(E) s_1 \text{else } s_2, Q) &= \\
&\quad (E \Rightarrow \text{wp}(s_1, Q) \wedge \neg E \Rightarrow \text{wp}(s_2, Q)) \\
\text{wp}(\text{nop}, Q) &= Q
\end{aligned}$$

Goal: show the Hoare triple is valid

Assignment Project Exam Help

1) Compute $\text{wp}(\text{prog}, \text{postcondition})$

$$\text{wp}(x := x + 1; y := x * (x + 5), y > 0)$$

$$= \text{wp}(x := x + 1, \text{wp}(y := x * (x + 5), y > 0))$$

$$= \text{wp}(x := x + 1, x * (x + 5) > 0))$$

$$= (x + 1) * (x + 6) > 0$$

2) Show the precondition implies wp

$$(x > 0) \Rightarrow ((x + 1) * (x + 6) > 0)$$

```

{ x=a }
if (a<0) x := -a;
{ x=|a| }

```

$$\begin{aligned}
\text{wp}(x := e, Q) &= Q[x \leftarrow e] \\
\text{wp}(s_1 ; s_2, Q) &= \text{wp}(s_1, \text{wp}(s_2, Q)) \\
\text{wp}(\text{if}(E) s_1 \text{else } s_2, Q) &= \\
&\quad (E \Rightarrow \text{wp}(s_1, Q) \wedge \neg E \Rightarrow \text{wp}(s_2, Q)) \\
\text{wp}(\text{nop}, Q) &= Q
\end{aligned}$$

Goal: show the Hoare triple is valid

[Assignment](#) [Project](#) [Exam](#) [Help](#)

1) Compute $\text{wp}(\text{prog}, \text{postcondition})$

$$\text{wp}(\text{if } (a < 0) \text{ x}:=-a, x = |a|)$$

$$= (a < 0 \Rightarrow \text{wp}(x := -a, x = |a|)) \wedge$$

$$(a \geq 0 \Rightarrow \text{wp}(\text{nop}, x = |a|))$$

$$= (a < 0 \Rightarrow -a = |a|) \wedge (a \geq 0 \Rightarrow x = |a|)$$

$$= (a \geq 0 \Rightarrow x = |a|)$$

2) Show the precondition implies wp

$$(x = a) \Rightarrow (a \geq 0 \Rightarrow x = |a|)$$

Dafny: A Verification Tool

<https://rise4fun.com/Dafny>

Assignment Project Exam Help

Microsoft
Research

dafny

Is this program correct?

```
1 method Main() {
2   print "hello, Dafny\n";
3   assert 10 < 2;
4 }
5 |
```

Example

```
1 function abs(x: int): int
2 {
3     if x < 0 then -x else x
4 }
```

```
{ true}
if (x<0)    r:=-x;
else         r:=x;
{ r=|x| }
```

Assignment Project Exam Help

```
5
6 method foo (x: int) returns (y: int)
7     requires true
8     ensures r == abs(x)
9 {
10     if x < 0 {      Add WeChat powcoder
11         r := -x;
12     }
13     else {
14         r := x;
15     }
16 }
```

https://powcoder.com

Observation: program verification is systematic and automatic if there is no loop!

Example

```
1 method foo (n: int) r
2   requires n >= 0
3   ensures r == 2*n
4 {
5   r := 0;
6   var i := 0; https://powcoder.com
7   while i < n
8   {
9     r := r+2; Add WeChat powcoder
10    i := i+1;
11  }
12 }
```

```
{ n ≥ 0 }
r:=0, i:=0;
while (i<n) {
  r := r+2;
  i++;
}
{ r = 2×n }
```

Assignment Project Exam Help

Add WeChat powcoder

		Description
☒	1	A postcondition might not hold on this return path.
	2	This is the postcondition that might not hold.

Loops $\{P\} \text{while } (E) \; s \; \{Q\}$

What is the WP?

Assignment Project Exam Help

Let $W = \text{while } (E) \; s$, then $\{P\} \text{while } (E) \; s \; \{Q\}$

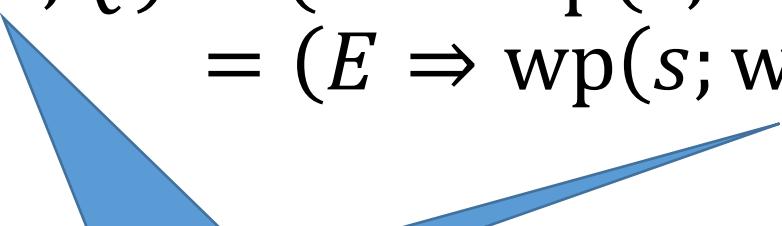
<https://powcoder.com>

is the same as $\{P\} \text{if } (E) \; s; W \text{ else nop } \{Q\}$

Add WeChat powcoder

By if-rule,

$$\begin{aligned}\text{wp}(W, Q) &= (E \Rightarrow \text{wp}(s; W, Q) \wedge \neg E \Rightarrow Q) \\ &= (E \Rightarrow \text{wp}(s; \text{wp}(W, Q))) \wedge \neg E \Rightarrow Q)\end{aligned}$$



Loop Invariant

Loop Invariant $\{P\}\text{while } (E) \; s \;\{Q\}$

$$Inv \Rightarrow (E \Rightarrow \text{wp}(s, Inv) \wedge \neg E \Rightarrow Q)$$

Hence, $Inv \wedge E \Rightarrow \text{wp}(s, Inv)$ and $Inv \wedge \neg E \Rightarrow Q$

(Proof is beyond the scope of this lecture)

<https://powcoder.com>

Loop invariant (Inv) is a proposition that is:

- 1) Initially true ($P \Rightarrow Inv$)
- 2) True after each iteration ($Inv \wedge E \Rightarrow \text{wp}(s, Inv)$)
- 3) Termination of loop implies the postcondition
 $(Inv \wedge \neg E \Rightarrow Q)$

Loop Invariant and Induction

Loop invariant (Inv) is a proposition that is:

- 1) Initially true ($P \Rightarrow Inv$)
- 2) True after each iteration ($Inv \wedge E \Rightarrow \text{wp}(s, Inv)$)
- 3) Termination of loop implies the postcondition
($Inv \wedge \neg E \Rightarrow Q$)

Intuitively, we are proving the correctness of an arbitrary number of loop iterations, by **induction!**

Example

Goal: show the Hoare triple is valid

Assignment Project Exam Help
1) Write down a tentative loop invariant (Inv)

$$r = 2 \times i \wedge i \leq n \quad \text{https://powcoder.com}$$

2) Show Inv is a loop invariant

Add WeChat powcoder

- $\{n \geq 0\} \ r := 0, i := 0; \{Inv\}$ is valid
- $Inv \wedge i < n \Rightarrow \text{wp}(r := r + 2; i++, Inv)$
- $Inv \wedge i \geq n \Rightarrow r = 2 \times n$

```
{ n ≥ 0 }  
r := 0, i := 0;  
while (i < n) {  
    r := r + 2;  
    i++;  
}  
{ r = 2 × n }
```

35

Assignment Project Exam Help
Program Verification
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Example

Goal: show the Hoare triple is valid

Assignment Project Exam Help

1) Write down a tentative loop invariant (Inv)

$$r = 2 \times i \wedge i \leq n$$

2) Show Inv is a Loop invariant

- $\{n \geq 0\} r:=0, i=0; \{Inv\}$ is valid

- $Inv \wedge i < n \Rightarrow wp(r:=r+2; i++, Inv)$

- $Inv \wedge i \geq n \Rightarrow r = 2 \times n$

```
{ n ≥ 0 }  
r := 0, i := 0;  
while (i < n) {  
    r := r + 2;  
    i++;  
}  
{ r = 2 × n }
```

Example

```
1 method f (n: int) returns (r: int)
2   requires n >= 0
3   ensures r == 2*n
4 {
5   r := 0;
6   var i := 0;
7   while i < n
8     invariant r == 2*i
9     invariant i <= n
10  {
11    r := r+2;
12    i := i+1;
13  }
14 }
```

```
{ n ≥ 0 }
r:=0, i:=0;
while (i<n) {
  r := r+2;
  i++;
}
{ r = 2×n }
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Example

Goal: show the Hoare triple is valid

Assignment Project Exam Help

1) Write down a tentative loop invariant (Inv)

$$r = \prod_{j=i+1}^n j \wedge i \geq 0 \wedge n \geq 0$$

2) Show Inv is a loop invariant

- $\{n \geq 0\} r:=1, i=n; \{Inv\}$ is valid
- $Inv \wedge i > 0 \Rightarrow wp(r:=r*i; i--; Inv)$
- $Inv \wedge i \leq 0 \Rightarrow r = n!$

```
{ n ≥ 0 }  
r := 1, i := n;  
while (i > 0) {  
    r := r * i;  
    i --;  
}  
{ r = n! }
```

Example

```
1 function fact (n: nat): nat
2 {
3     if n == 0 then 1 else
4     if n == 1 then 1 else
5         n * fact (n-1)
6 } Assignment Project Exam Help
7
```

```
8 method f (n: nat) returns (r: nat)
9     requires n >= 0
10    ensures r == fact(n)
```

```
11 {
12     var i := n;
13     r := 1;
14     while i > 0
15         invariant i >= 0
16         invariant n >= 0
17         invariant r * fact(i) == fact(n)
18     {
19         r := r * i;
20         i := i - 1;
21     }
22 }
```

```
{ n ≥ 0 }
r:=1, i:=n;
while (i>0) {
    r := r*i;
    i --;
}
{ r = n! }
```

<https://powcoder.com>

Add WeChat powcoder

Verification in Practice

Goal: show the Hoare triple is valid

- 1) Write down a tentative loop invariant (Inv)
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- 2) Show Inv is a loop invariant
<https://powcoder.com>

Step 2) is automatic, but 1) is mostly manual ...

Significant artifacts (e.g., simple OS) have been verified, but with pains (e.g., 3 person-years)

Total vs. Partial Correctness

$\{P\}$ while (E) $s \{Q\}$

Is this program correct?

Assignment Project Exam Help

```
{a ≥ 0, epsilon > 0}
float sqrt (float a, float epsilon) {
    float x := 1.0;
    while (x*xAddWeChat powcoder.com < a-epsilon)
        x := x;
    return x;
}
{a - epsilon ≤ x² ≤ a + epsilon}
```

We can verify it with invariant $\{true\}$. Why?

Total vs. Partial Correctness

$\{P\}\text{while } (E) \; s \; \{Q\}$

Partial correctness: if the loop terminates, Q must be true. *Assignment Project Exam Help*
However, the loop might not terminate

E.g., $\{P\}\text{while } (\text{true}) \; s \{Q\}, Inv \wedge \text{true} \Rightarrow Q \text{ is true}$

The loop invariant only guarantees partial correctness

Total correctness: prove loop determinates
(undecidable in general)

Summary

Goal: prove a program s is correct

Step 1: formalize “correctness” by writing down the precondition P and postcondition Q

Step 2: show that the Hoare triple $(\{P\} s \{Q\})$ is valid

- Mostly automatic, except for loops

What is verified?

Given any state satisfying P , the final state after executing s must satisfy Q , if s terminates

#36

Assignment Project Exam Help
OOP and Garbage Collection

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

HW7

- Last assignment
- Due on the last day of class (Dec. 11) midnight
(no late submission)

<https://powcoder.com>

Add WeChat powcoder

Abstract Data Types

Primitive types: values and operations on values

User-defined types: records, lists

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Focus on values

<https://powcoder.com>

ADT: defined by a set of operations on a type

[Add WeChat powcoder](#)

Focus on operation

Stack is a type with new, pop, push, empty ...

Internal representation is less relevant

Classifying Operations

Creators: create new objects of type

Assignment Project Exam Help

Producers: create new objects from old ones
<https://powcoder.com>

Mutators: change objects, e.g., `list.add(n)`

Observers: take objects of ADT and return
objects with different type, e.g., `list.size()`

ADT Example

int

Assignment Project Exam Help

Creators: numeric literals 1, 2, 3, ...
<https://powcoder.com>

Producers: arithmetic operations +, -, *, /, ...
Add WeChat powcoder

Observers: comparison operators ==, !=, <, >

Mutators: none (immutable)

ADT Example

List

Assignment Project Exam Help

Creators: ArrayList, LinkedList, ...
<https://powcoder.com>

Producers: Collections.unmodifiableList()

Observers: size(), get()

Mutators: add(), remove(), ...

ADT Example

String

Assignment Project Exam Help

Creators: String(), String(char[])
<https://powcoder.com>

Producers: concat(), substring(), ...

Observers: length(), charAt(), ...

Mutators: none (immutable)

OOP Terminology

Class: *a richer version of ADT*

Object (Instance): ~~Assignment Project Exam Help~~ a variable of a class (a value of a type)

Field: variable in a class
<https://powcoder.com>

Method: operation in a class
~~Add WeChat powcoder~~

Object-Oriented Programming

Key elements:

- Encapsulation
- Subtyping
- Inheritance

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Encapsulation (Information Hiding)

- Group data and operations in one place (typically, in one class)
- Hide irrelevant details (using visibility modifiers, such as *public*, *private*, *protected*)

Add WeChat powcoder

Subtyping

Sometimes, every value of type B is of type A

- e.g., a `Rectangle` is always a `Shape`

<https://powcoder.com>

We say B is a *subtype* of A, meaning

"every object that satisfies interface of B also satisfies interface of A"

[Add WeChat powcoder](#)

Subtyping

```
interface Shape {  
    public double area();  
    public int edges();  
}
```

Assignment Project Exam Help

<https://powcoder.com>

```
class Circle implements Shape {  
    double radius;  
    public double area() { return 3.14*radius*radius; }  
    public int edges() { return 1; }  
}
```

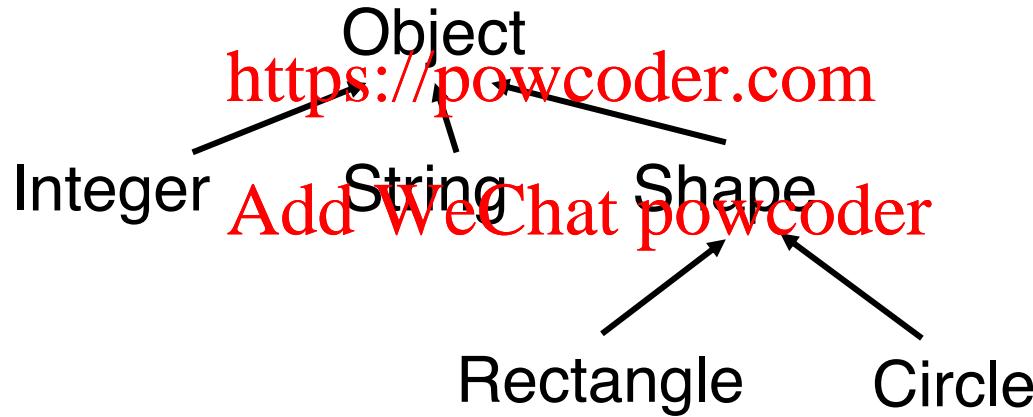
Add WeChat powcoder

Subtyping Relation

We write $A \leq B$ to mean A is a subtype of B

Relation \leq defines a partial ordering on types

Assignment Project Exam Help



Reflexivity: $\forall T. T \leq T$

Antisymmetry: $\forall T_1, T_2 . T_1 \leq T_2 \wedge T_2 \leq T_1 \Rightarrow T_1 = T_2$

Transitivity: $\forall T_1, T_2, T_3 . T_1 \leq T_2 \wedge T_2 \leq T_3 \Rightarrow T_1 \leq T_3$

Subtype Polymorphism

“Casting” a subtype to its supertype is ALWAYS safe (known as “Upcast”)

Assignment Project Exam Help

Hence, a function with parameter of type B can take any data with type $A \leq B$

Add WeChat [powcoder](https://powcoder.com)

```
int f (Object o) {...} // takes object of any type
int g (Shape s) {...} // takes object of subtype
                      // of Shape, such as Circle
```

Upcasting and Downcasting

- Upcast: change to a supertype
 - Always safe
- Downcast: change to a subtype
 - Not always safe

```
Rectangle r = new Rectangle(2,3); // OK  
Shape s = r; // Upcast is always safe  
r = (Rectangle) s; // OK, but downcast  
// is not always safe
```

Java checks type casts at run time

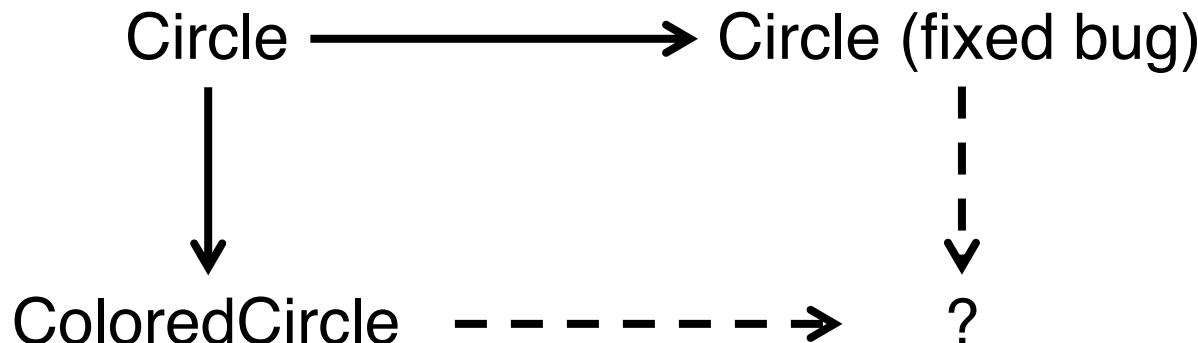
Motivation of Inheritance

```
class Circle implements Shape {  
    double radius;  
    public double area() {return 3.14*radius*radius};  
    public int sides() {return 1};  
}
```

<https://powcoder.com>

How to implement ColoredCircle?

Option 1: copy-and-paste
[Add WeChat powcoder](#)



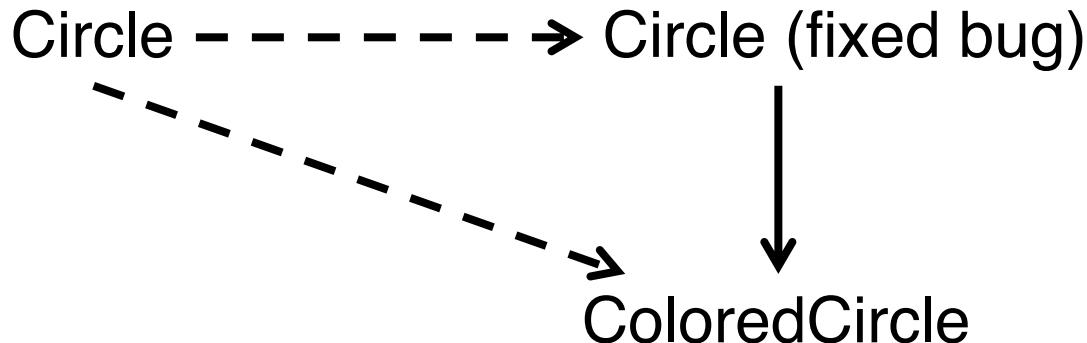
Inheritance

```
class ColoredCircle extends Circle {  
    private Color color;  
    ...  
    Color getColor {return color;}  
    // methods area, edges are inherited from Circle  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Inheritance

```
class ColoredCircle extends Circle {  
    private Color color;  
    ...  
    Color getColor {return color;}  
    // methods area, edges are inherited from Circle  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Inheritance introduces subtyping:

ColoredCircle **inherits all fields & methods**

Circle **is the supertype of** ColoredCircle

ColoredCircle **is the subtype of** Circle

Overriding

```
class DoubleCircle extends Circle {  
    public DoubleCircle(double r) {super(r);}  
    public int areas() {return 2;  
}
```

<https://powcoder.com>

Add WeChat powcoder
Subclass may redefine methods in super class

Object-Oriented Programming

Key elements:

- Encapsulation [Assignment](#) [Project](#) [Exam](#) [Help](#)
- Subtyping <https://powcoder.com>
- Inheritance [Add WeChat powcoder](#)

How are these features implemented?

Running Example

Assignment Project Exam Help
Circle: edges return 1

subtype of

<https://powcoder.com>

subtype of

ColoredCircle: inherits edges
has a new filed “color”

Add WeChat powcoder

DoubleCircle: edges return 2

Memory Layout (Fields)

An object has

- Fields (and ones from super class)
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Circle object1:

DoubleCircle object:

ColoredCircle object:

<https://powcoder.com>

Add WeChat powcoder

radius

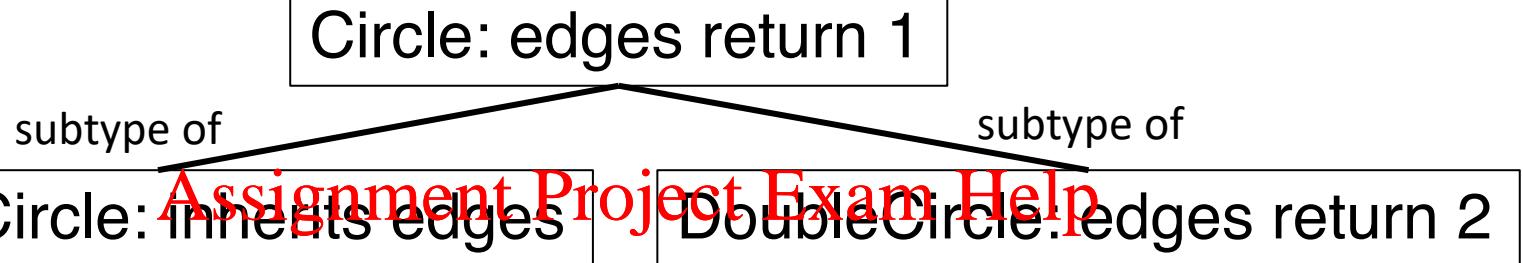
radius

radius
color

```
foo (Circle s) {  
    s.radius; // offset?  
}
```

Caution: new fields in subtype should extend inherited fields due to subtype polymorphism

Memory Layout (Functions)



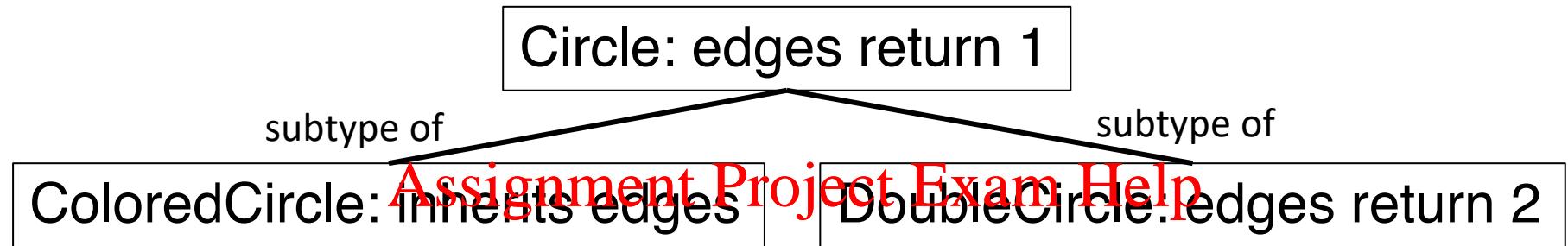
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      // Which implementation?  
}
```

Static dispatch: s.edges() always returns 1

Dynamic dispatch: return value of s.edges() controlled by the type of s

Static Dispatch



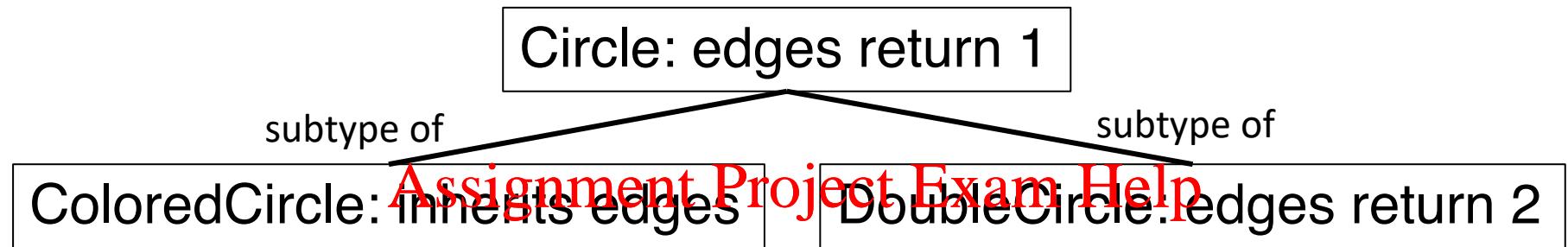
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      /Add WeChat powcoder  
}
```

Dispatch to the implementation in class Circle

Hence, `s.edges()` always returns 1

Implementation: Static



<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      /Add WeChat powcoder  
}
```

The compiler can always tell which implementation at compile time (e.g., the `edges` method in class `Circle`)

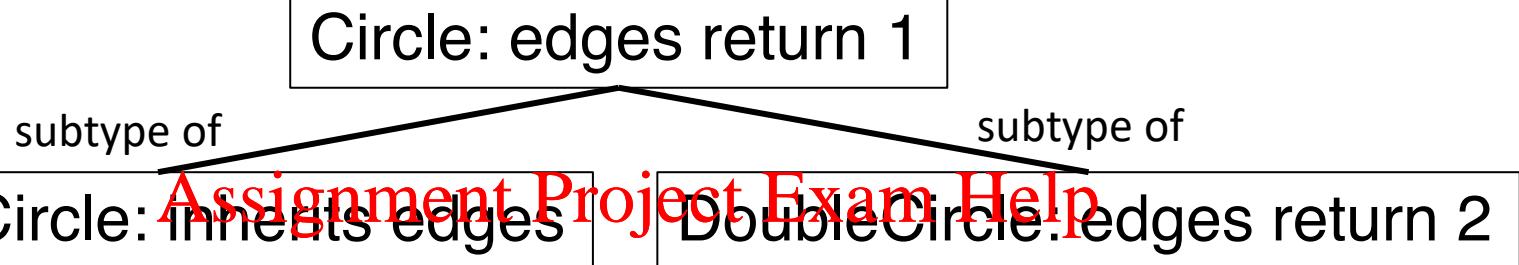
#31

Assignment Project Exam Help
OOP and Garbage Collection
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Memory Layout (Functions)



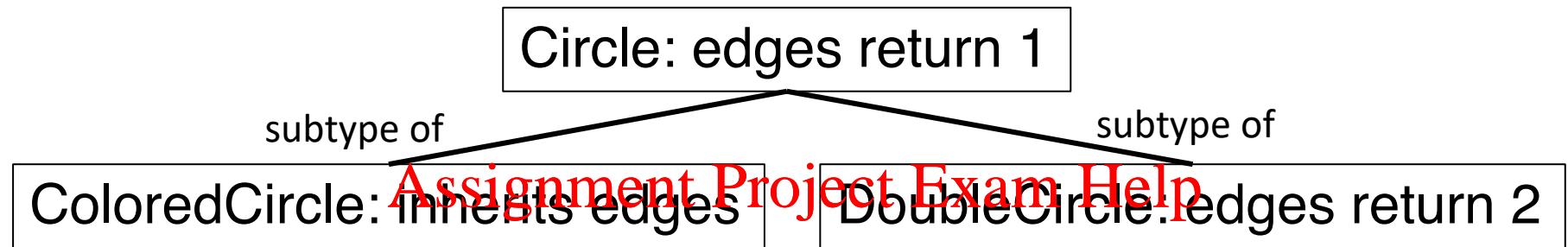
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      // Which implementation?  
}
```

Static dispatch: s.edges() always returns 1

Dynamic dispatch: return value of s.edges() controlled by the type of s

Implementation: Static

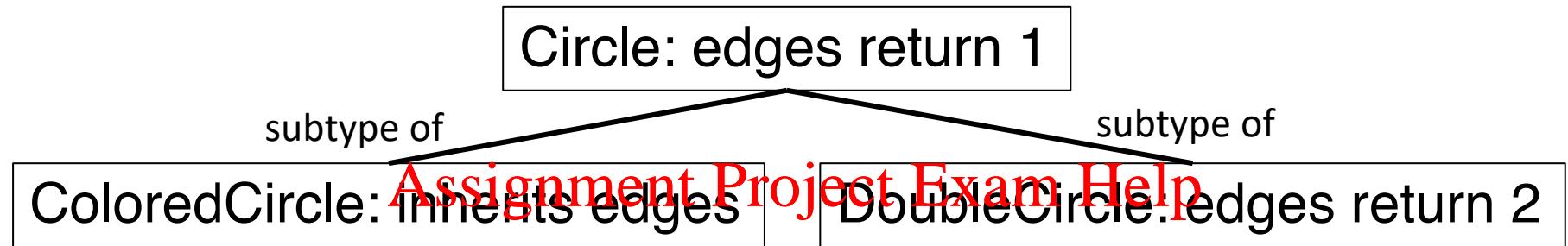


<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      /Add WeChat powcoder  
}
```

The compiler can always tell which implementation at compile time (e.g., the `edges` method in class `Circle`)

Dynamic Dispatch



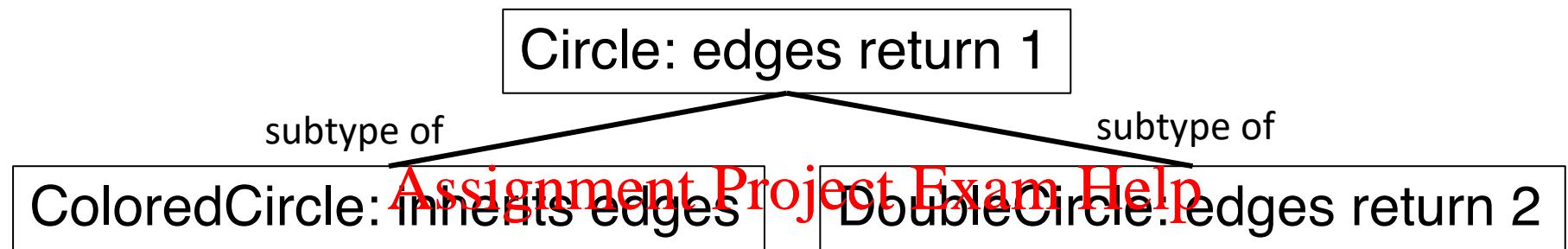
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      // Which implementation?  
}
```

Dispatch to the implementation based on the type of the object s

Hence, `s.edges()` returns 2 when `s` is an object of class `DoubleCircle`

Implementation: Dynamic



<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      //Add WeChat powcoder  
}
```

The compiler does not know the type of s. How can it dispatch the method call to the correct implementation?

Trivial Memory Layout

An object has

- Fields (and ones from super class)
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Methods (and ones from super class)
<https://powcoder.com>

Circle object1:

radius
edges: binary
area: binary

Circle object2:

radius
edges: binary
area: binary

DoubleCircle object:

radius
edges': binary
area: binary

Issues: each object has a copy of impl. code (waste space)
polymorphic functions need to distinguish different
layouts of classes (to find method offset)

Virtual Table (vtable)

- Tentative design

A (shared) table containing method binaries

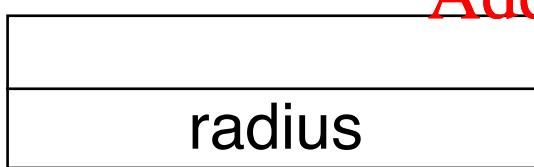
Assignment Project Exam Help

To save memory: one table per **Class**

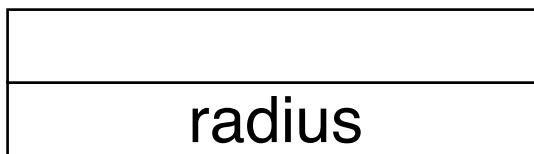
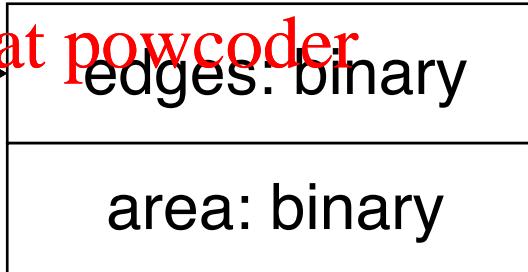
<https://powcoder.com>

Circle's vtable

Circle objects:



Add WeChat powcoder



Fields stored in each object

Points to methods stored in one shared vtable

Virtual Table (vtable)

- Tentative design

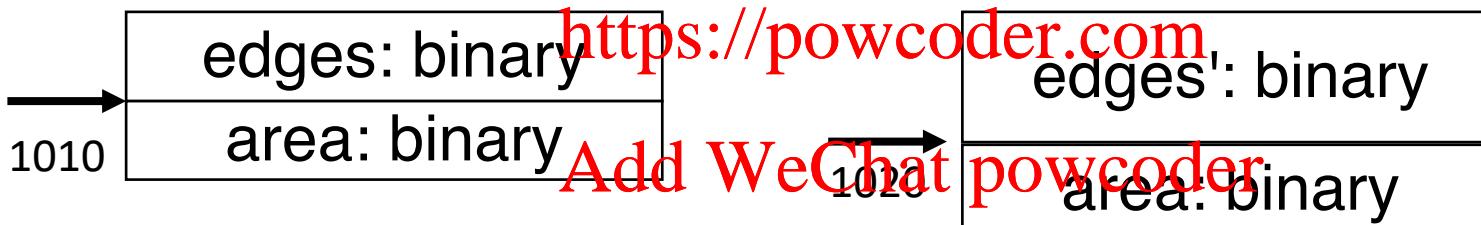
A (shared) table containing methods

Override?

Assignment Project Exam Help

Circle's vtable

DoubleCircle's vtable



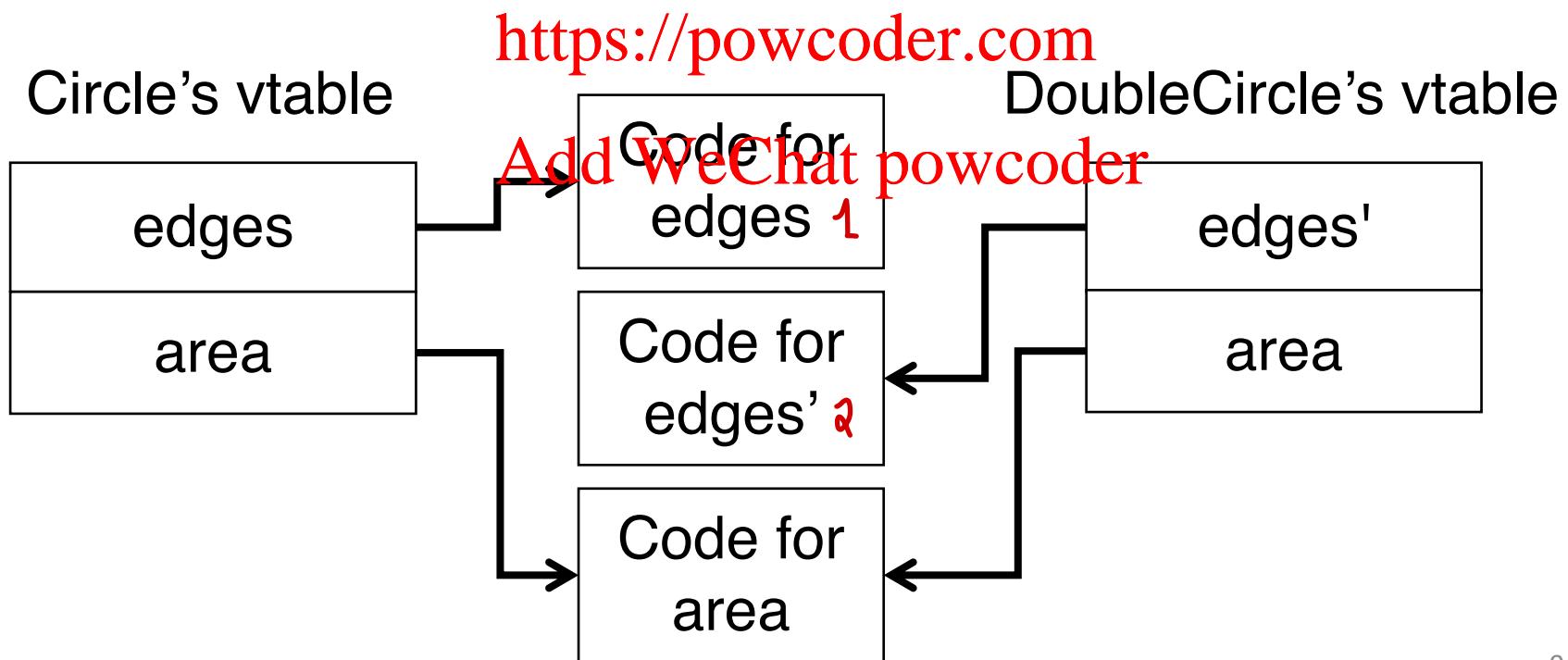
```
foo (Circle s) {  
    s.area(); // code has different offsets  
}
```

Issue: foo is compiled to different binaries with different offsets for different types of s

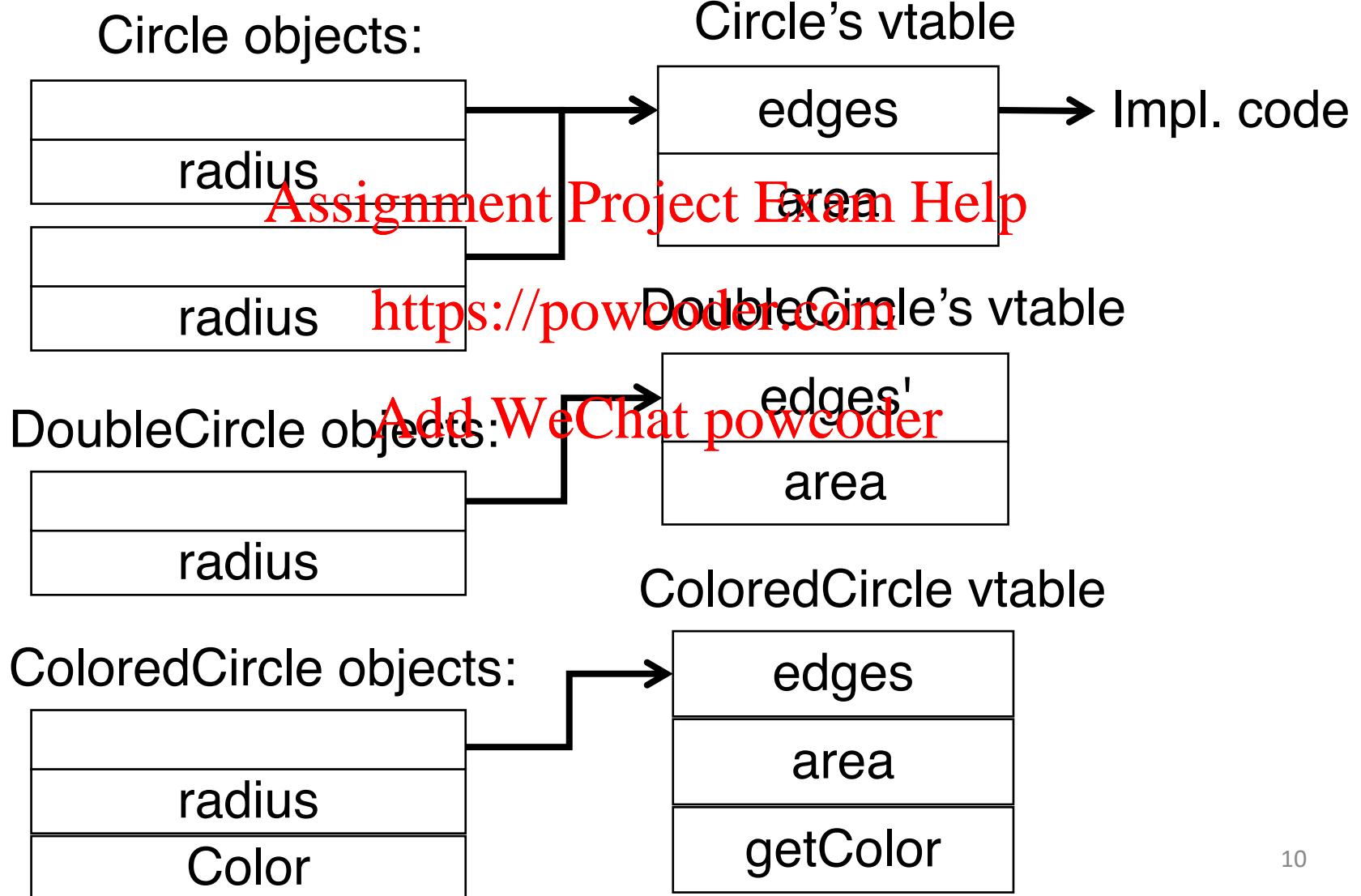
Virtual Table (vtable)

A (shared) table containing **pointers** to methods

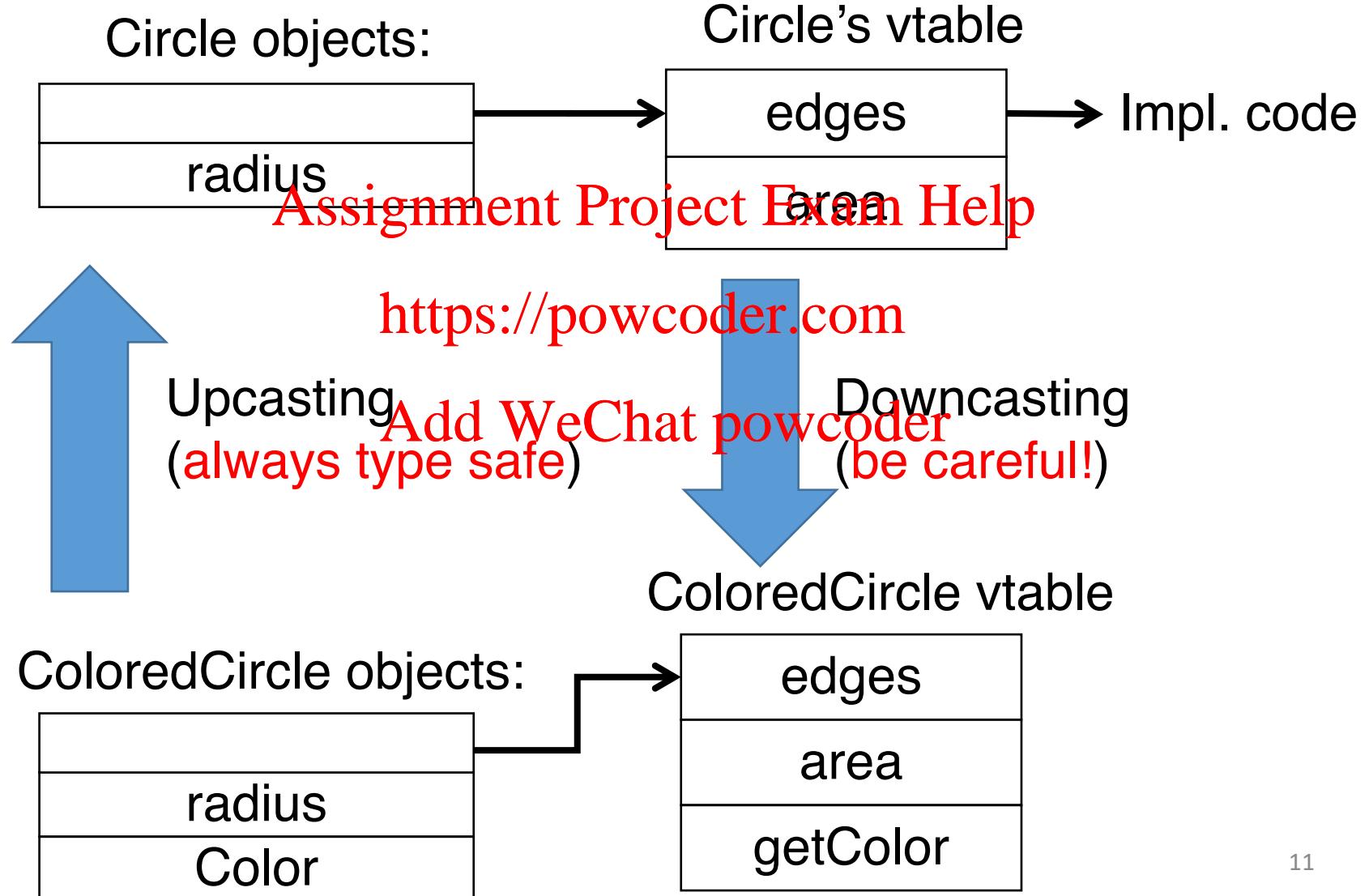
Assignment Project Exam Help



Virtual Table (vtable)



Downcasting/Upcasting



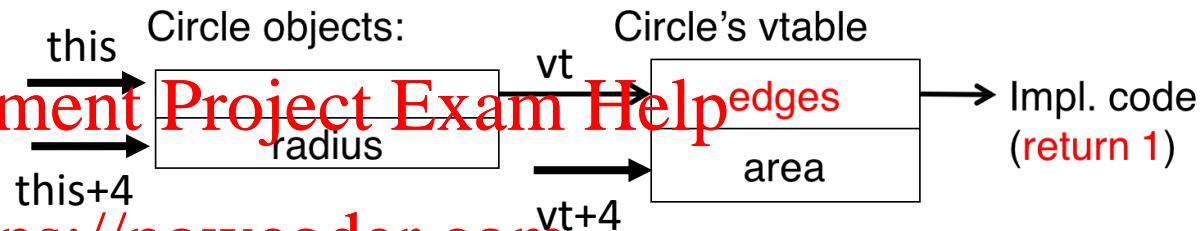
Member Lookup: Case 1

When s is an object of class Circle

```
foo (Circle s) {  
    s.radius;  
    s.area();  
    s.edges();  
}
```

Assignment Project Exam Help

<https://powcoder.com>



Add WeChat powcoder

```
foo (Circle s) {  
    vt = *this;  
    *(this+4); //value of radius  
    call *(vt+4); // method area  
    call *vt; // method edges  
}
```

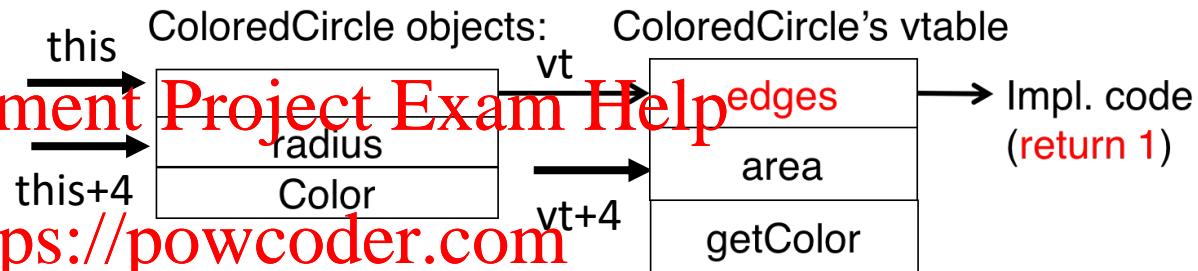
s.edges returns 1

Member Lookup: Case 2

When s is an object of class ColoredCircle

```
foo (Circle s) {  
    s.radius;  
    s.area();  
    s.edges();  
}
```

Assignment Project Exam Help
<https://powcoder.com>



Add WeChat powcoder

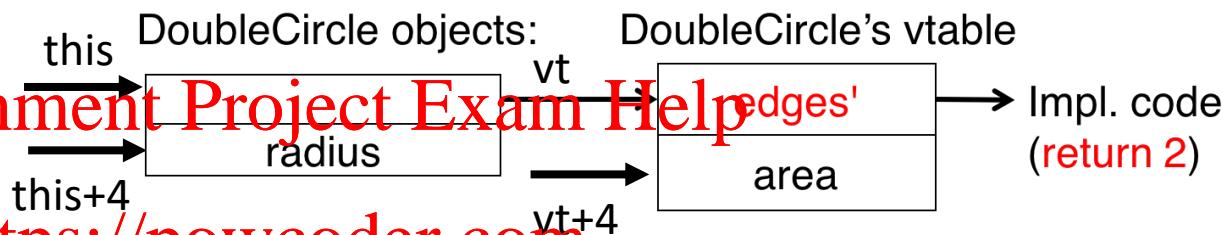
```
foo (Circle s) {  
    vt = *this;  
    *(this+4); //value of radius  
    call *(vt+4); // method area  
    call *vt; // method edges  
}
```

s.edges returns 1
Upcasting is type safe

Member Lookup: Case 3

When s is an object of class DoubleCircle

```
foo (Circle s) {  
    s.radius;  
    s.area();  
    s.edges();  
}
```



Assignment Project Exam Help

<https://powcoder.com>



Add WeChat powcoder

```
foo (Circle s) {  
    vt = *this;  
    *(this+4); //value of radius  
    call *(vt+4); // method area  
    call *vt; // method edges  
}
```

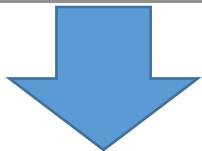
s.edges returns 2

Dynamic Dispatch with VTables

```
foo (Circle s) {  
    s.radius;  
    s.area();  
    s.edges();  
}
```

Assignment Project Exam Help

<https://powcoder.com>



Add WeChat powcoder

```
foo (Circle s) {  
    vt = *this;  
    *(this+4); //value of radius  
    call * (vt+4); // method area  
    call *vt; // method edges  
}
```

One implementation
for all subtypes!

Obj → vt → binary

Static vs. Dynamic Dispatching

Methods are dynamic

Assignment Project Exam Help

```
foo (Circle s) {  
    vt = *this;  
    *(this+4); //value of radius  
    call vt[4]; //method area  
    call *vt; // method edges
```

<https://powcoder.com>

```
foo (Circle s) {  
    s.radius;  
    s.area();  
    s.edges();  
}
```

Add WeChat powcoder

Methods are static

```
foo (Circle s) {  
    *(this+4); //value of radius  
    call Circle.area; // not in vt  
    call Circle.edges; // not in vt  
}
```

Cost of Dynamic Dispatch

Dynamic dispatch has costs, but is better for extensibility

Assignment Project Exam Help

In C++: static by default, except the virtual methods

<https://powcoder.com>

In Java: dynamic by default, except the ones that cannot be overridden (e.g., final and static methods)

In Python: all methods use dynamic dispatch

Static vs Dynamic Dispatch

```
foo (Circle s) {  
    s.edges(); // which implementation?  
}
```

~~Assignment Project Exam Help~~

Static dispatch: `s.edges()` always returns 1

Dynamic dispatch: return value of `s.edges()` controlled by
the type of `s`

~~Add WeChat powcoder~~

Static dispatch is easier to implement and more efficient
Dynamic dispatch less efficient, but provides better
extensibility (central to object-oriented programming)

CMPSC 461: Programming Language Concepts

Assignment 7. Due: Dec. 11, 11:59PM (**no late submission**)

For this assignment, you need to submit your solution to **Gradescope**. You may discuss the assignment with other students, but all submitted work must be your own work.

Problem 1 [8pt] Prove that the following two Hoare triples are valid. (Hint: in predicate logic $P_1 \Rightarrow P_2$ is equivalent to $\neg P_1 \vee P_2$).

a) (4pt)

$$\begin{array}{l} \{x = 1\} \\ y := x + 2; \\ y := y * 3; \\ \{y > 8\} \end{array}$$

b) (4pt)

$$\begin{array}{l} \{x = 2, y = 3\} \\ \textbf{if } (x < 1) \quad y := 5; \\ \textbf{else} \quad \quad \quad y := y - 1; \\ \{y = 2\} \end{array}$$

Assignment Project Exam Help

Problem 2 [12pt] Below is the pseudo code that computes $2x + 3y$, along with the pre- and post-conditions to be verified.

$$\begin{array}{l} \{z = 2x \wedge n = 0 \wedge y > 0\} \\ \textbf{while } (n < y) \{ \\ \quad z = z + 3; \\ \quad n = n + 1; \\ \} \\ \{z = 2x + 3y\} \end{array}$$

Add WeChat powcoder

Write down a loop invariant and prove the correctness of this program. That is, write down an invariant, and show that the loop invariant is 1) true before the first execution of the loop, 2) true after the execution of each loop iteration, given that the loop condition and the invariant are both true before the iteration, and 3) strong enough to prove the correctness of the code (i.e., $z = 2x + 3y$ is true after the loop, if it terminates).

Problem 3 [12pt] Consider the following Java classes:

```
class A {
    public int foo () { return 1; }
    public void message () { System.out.println( "A"); }
}
class B extends A {
    public void message () { System.out.println( "B"); }
}
class C extends A {
    public int foo() {return 3; }
}
class D extends C {
    public void message () {System.out.println( "D"); }
}
```

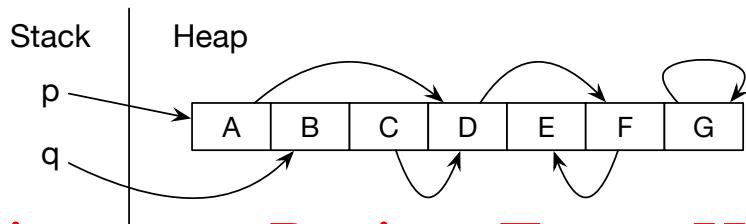
a) (8pt) Draw the virtual tables of classes A, B, C, D, and the code implementations that they point to.

b) (4pt) Consider the following function:

```
void func(C c) { c.message(); }
```

With subtype polymorphism, this function prints “A” given an instance of class C; prints “D” given an instance of class D. Give the pseudo code for the body of `func` after compilation to explain how does this happen with dynamic dispatching. Assume that the base address of `c` is stored in a pointer `this`, and that the target machine has 4-byte addresses. You can ignore the calling sequence (e.g., save and restore registers, store return address and so on).

Problem 4 [12pt] Consider the heap state as shown below before garbage collection:



Assignment Project Exam Help

a) (3pt) For the Mark-and-Compact algorithm, what objects remain on the heap after collection? Do they stay in the same location after collection?

b) (9pt) Consider algorithms Mark-and-Sweep, Mark-and-Compact, Stop-and-Copy. For each of them, write down the objects being visited during the collection in sequence. Assume 1) reachability analysis uses depth-first search, which will skip objects that have already been visited, 2) search starts from `p`, and then `q`, 3) when the entire heap is traversed, objects are visited from left to right. You don't need to write down the newly created object copies, if any.

<https://powcoder.com>

Add WeChat powcoder

Problem 5 [6pt] For each of the following data characteristics, **briefly explain** which of the garbage collection algorithms discussed in class (Mark and Sweep, Mark and Compact, Stop and Copy) would be the most appropriate and why. Make no other assumptions than those given.

a (3pt) All objects are of the same size and many have relatively long lifetime.

b (3pt) Most objects have short lifetime and data locality greatly improves application's performance.

38

Assignment Project Exam Help
OOP and Garbage Collection

<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Final

Dec. 16 (Wednesday), 6:50PM – 8:40PM

Cumulative, covers everything in the semester
Roughly a uniform distribution on course materials
30% of final grade

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Format: ZOOM-based exam, similar to Midterm 2

Next Friday: review for the final, HW7 due

Teaching Evaluation

Don't forgot the teaching evaluation

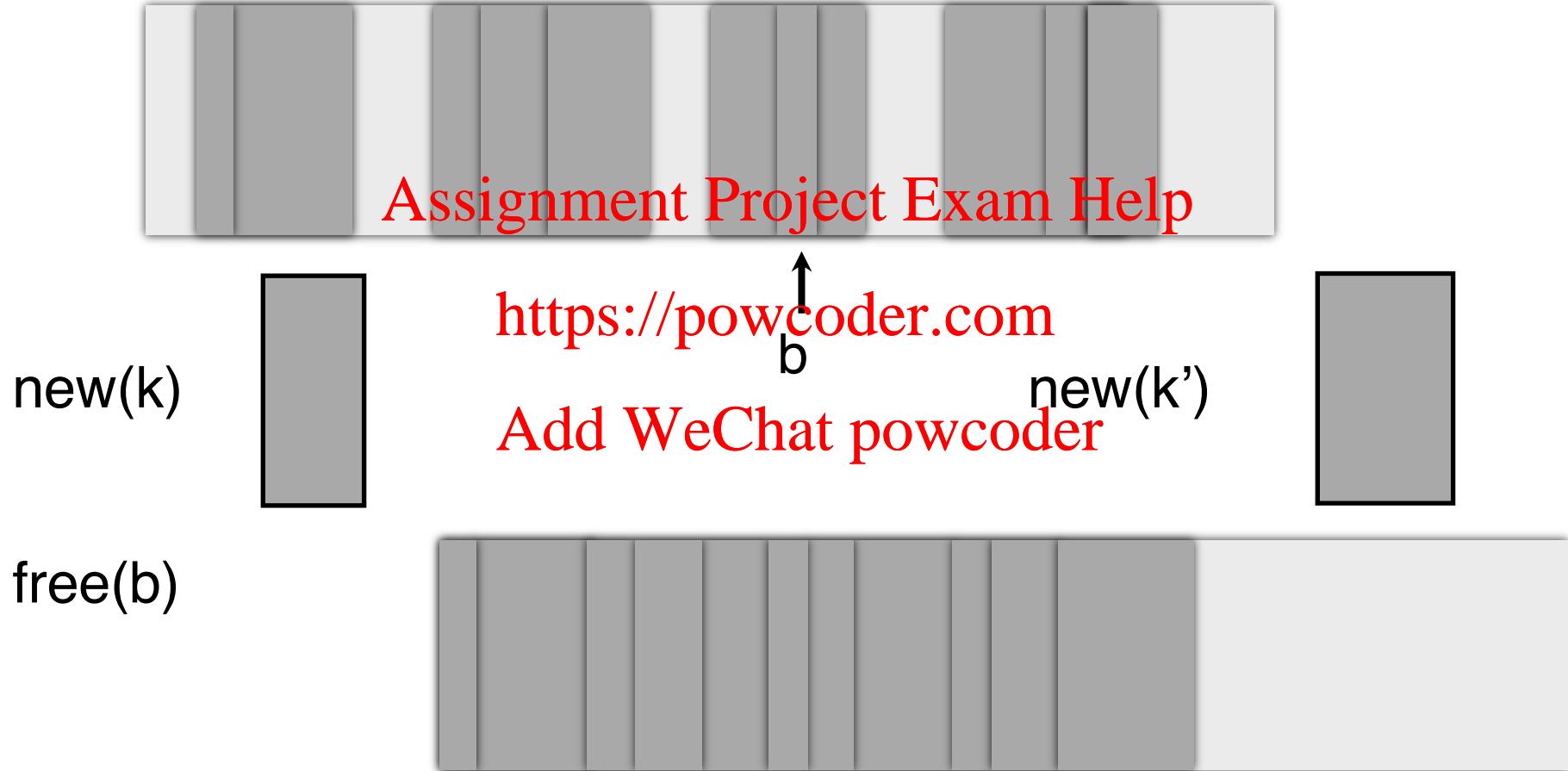
Assignment Project Exam Help

Feedbacks are appreciated

<https://powcoder.com>

Add WeChat powcoder

Heaps



Independent lifetimes of objects make heap management difficult

Heap Management

Allocator: a routine takes size of requested heap space, and search for free space

[Assignment](#) [Project](#) [Exam](#) [Help](#)

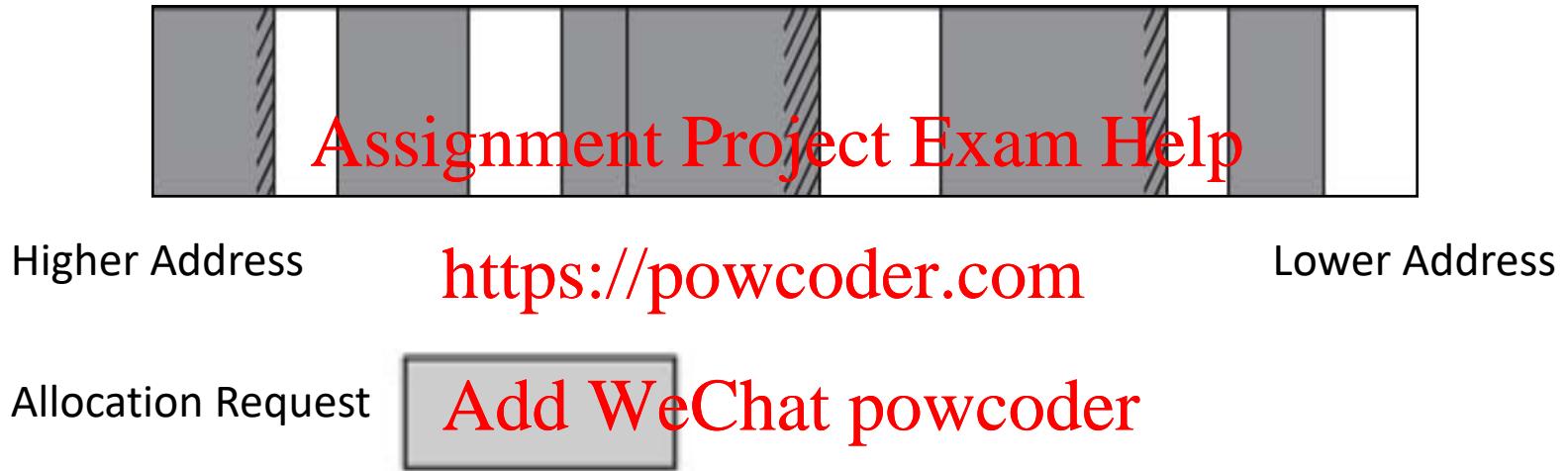
Usually, heap ~~is managed in blocks~~. Allocator may return larger block than requested

[Add WeChat](#) [powcoder](#)

Deallocator: collect free space and merge with other free space when possible

Heap compaction: move all used blocks to one end

Heap-Based Allocation



Internal fragmentation:
the allocation request is smaller than the assigned memory block

External fragmentation:
none of the scattered free space is large enough for the request

Heap Management Algorithms (not covered in this course)

First-fit: select the first free block that is large enough

Assignment Project Exam Help

Best-fit: select the smallest free block that fits

<https://powcoder.com>

Buddy system: maintain various pools of free blocks with size of 2^k

Fibonacci heap: maintain various pools of free blocks with size of Fibonacci numbers

Heap Management

Programmer Management (C, C++)

- Pros: implementation simplicity, performance
- Cons: error prone (dangling pointers, memory leaks)

<https://powcoder.com>

```
Node *p, *q;  
p = new Node();  
q = new Node();  
q = p; // memory leaks  
delete(p); // q becomes dangling pointer
```

Heap Management

Automatic Management (Java, Scheme)

- No dangling pointers, no memory leaks
- Cost: Slower than programmer management
<https://powcoder.com>

Add WeChat powcoder

Garbage Collection

Garbage: inaccessible heap objects

```
void foo () {  
    int* a = new int[10];  
    return;  
}
```

Issues of heap management.

- Collect too aggressively: dangling pointers
- Collect too conservatively: memory leaks
- Key problem: collect only objects that are ***inaccessible*** from program

GC I: Reference Counting

Maintain a reference count with each heap object

Set to 1 when object is created

<https://powcoder.com>

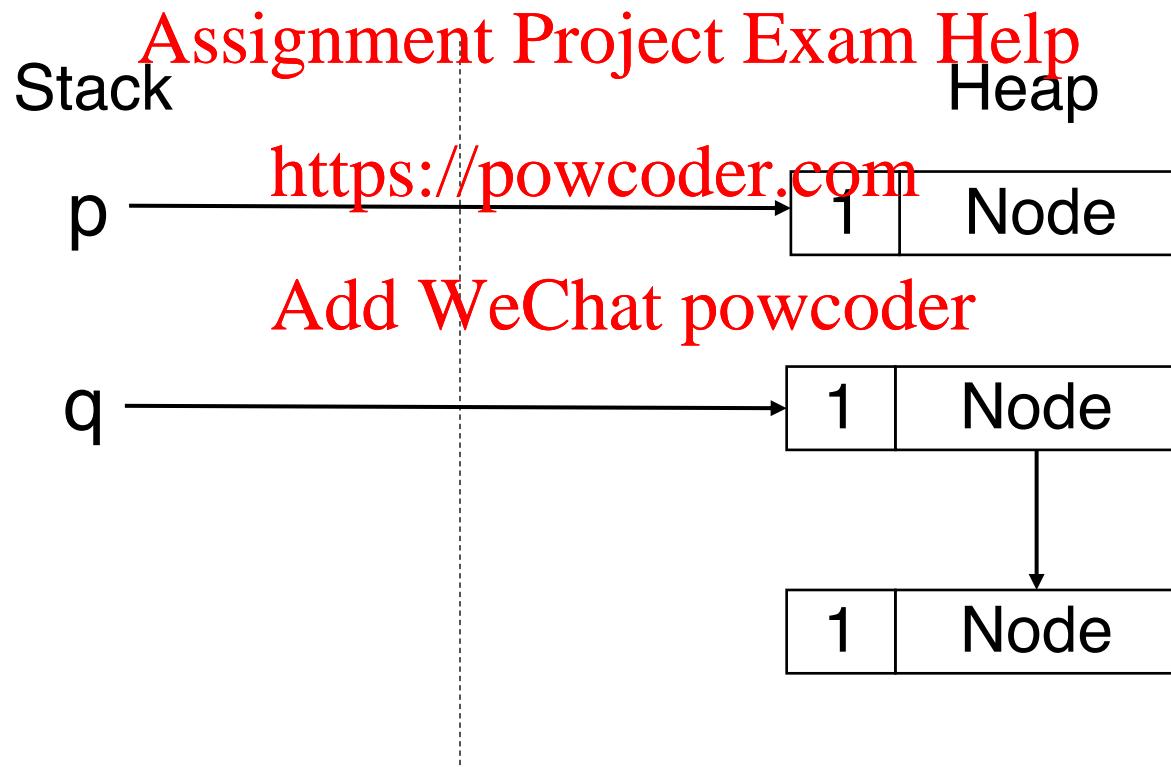
Incremented each time new reference to it is created

Add WeChat powcoder

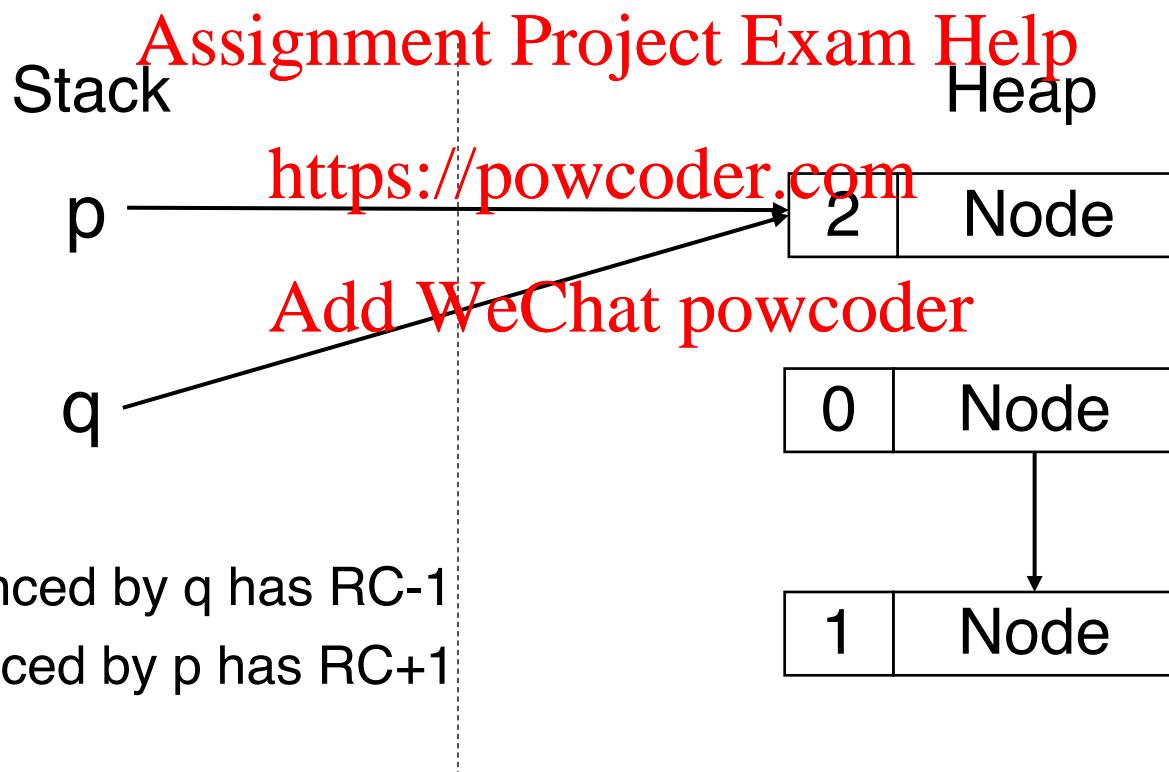
Decrement each time reference to it is deleted

Collect when count becomes 0

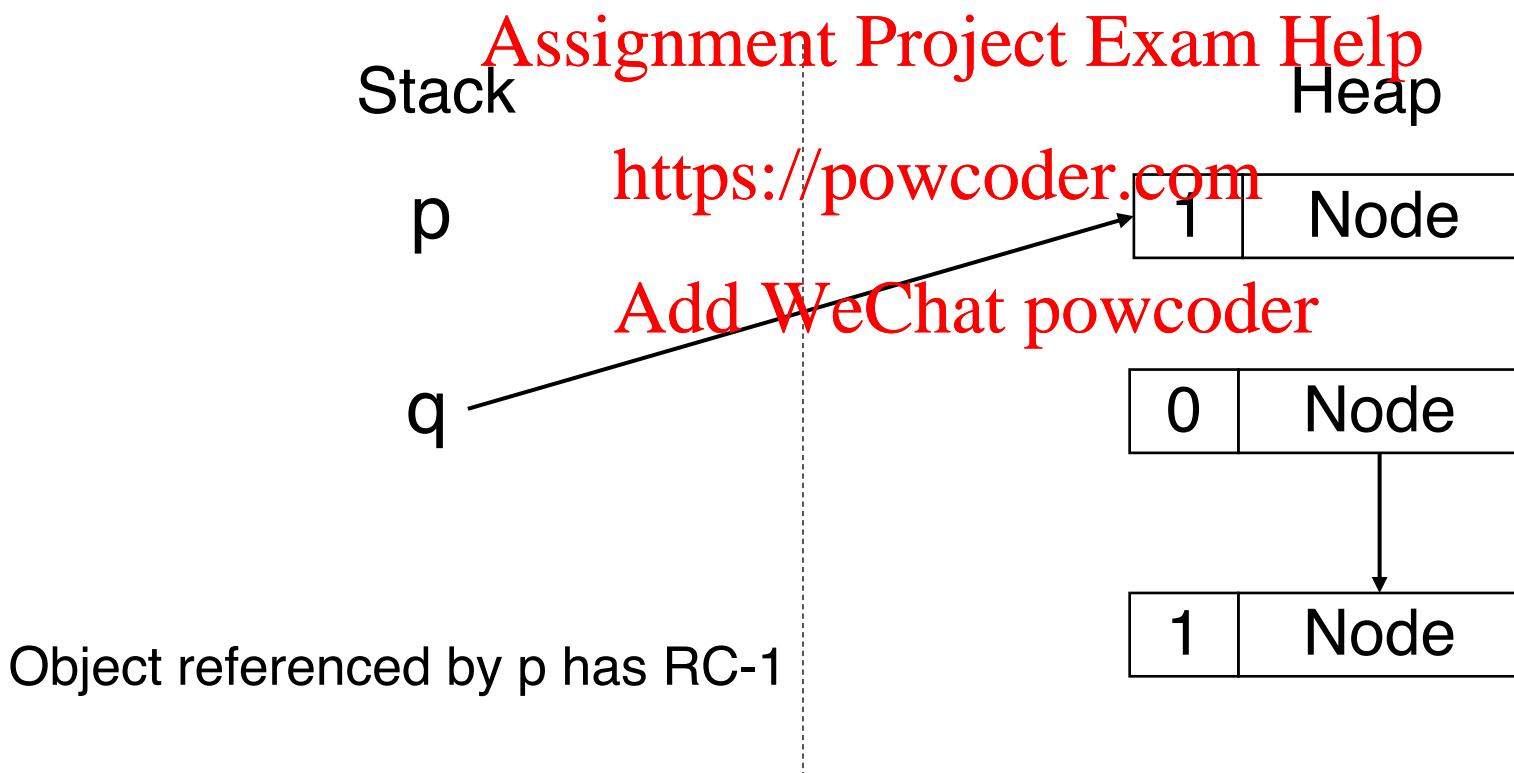
```
Node *p, *q;  
p = new Node();  
q = new Node();  
→ q.next = new Node();  
q = p;  
p = null;
```



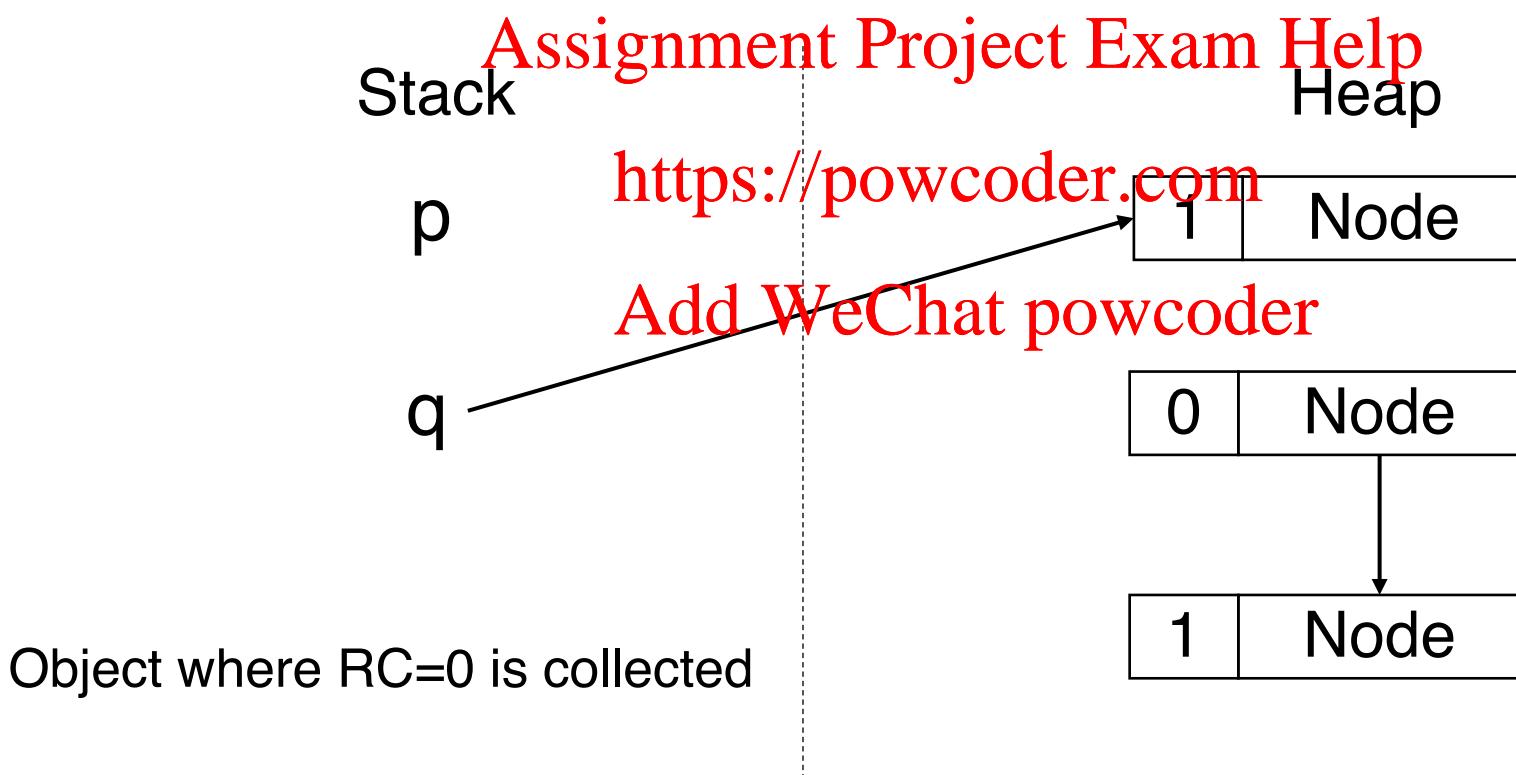
```
Node *p, *q;  
p = new Node();  
q = new Node();  
q.next = new Node();  
→ q = p;  
p = null;
```



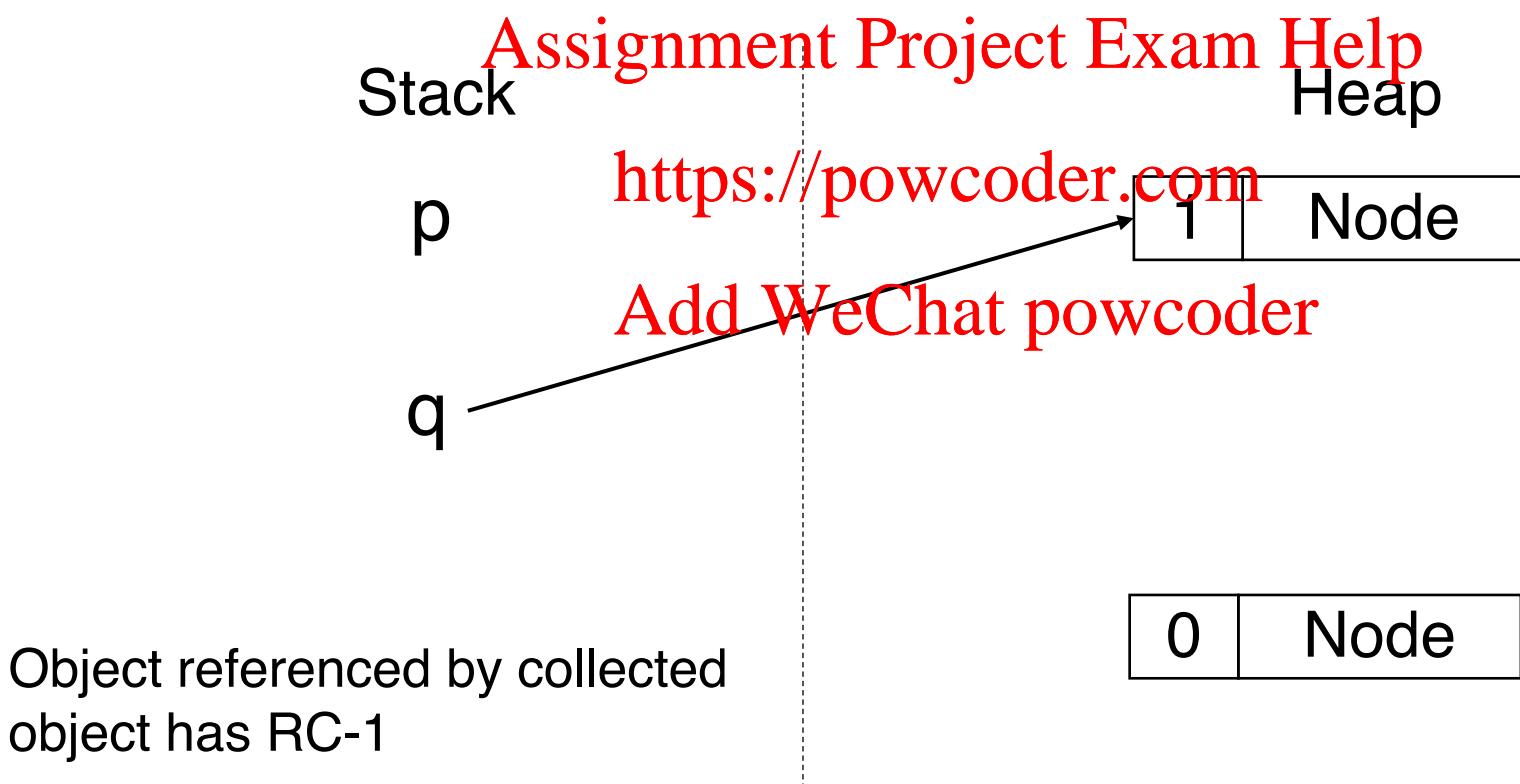
```
Node *p, *q;  
p = new Node();  
q = new Node();  
q.next = new Node();  
q = p;  
p = null;
```



Garbage Collection



Garbage Collection



GC I: Reference Counting

When is an object dereferenced?

- Reference is LHS of Assignment
- Reference on stack is destroyed when function returns
- Reference is destroyed when an object with count 0 is collected

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

GC I: Reference Counting

Reference Counting is about **Object Ownership**

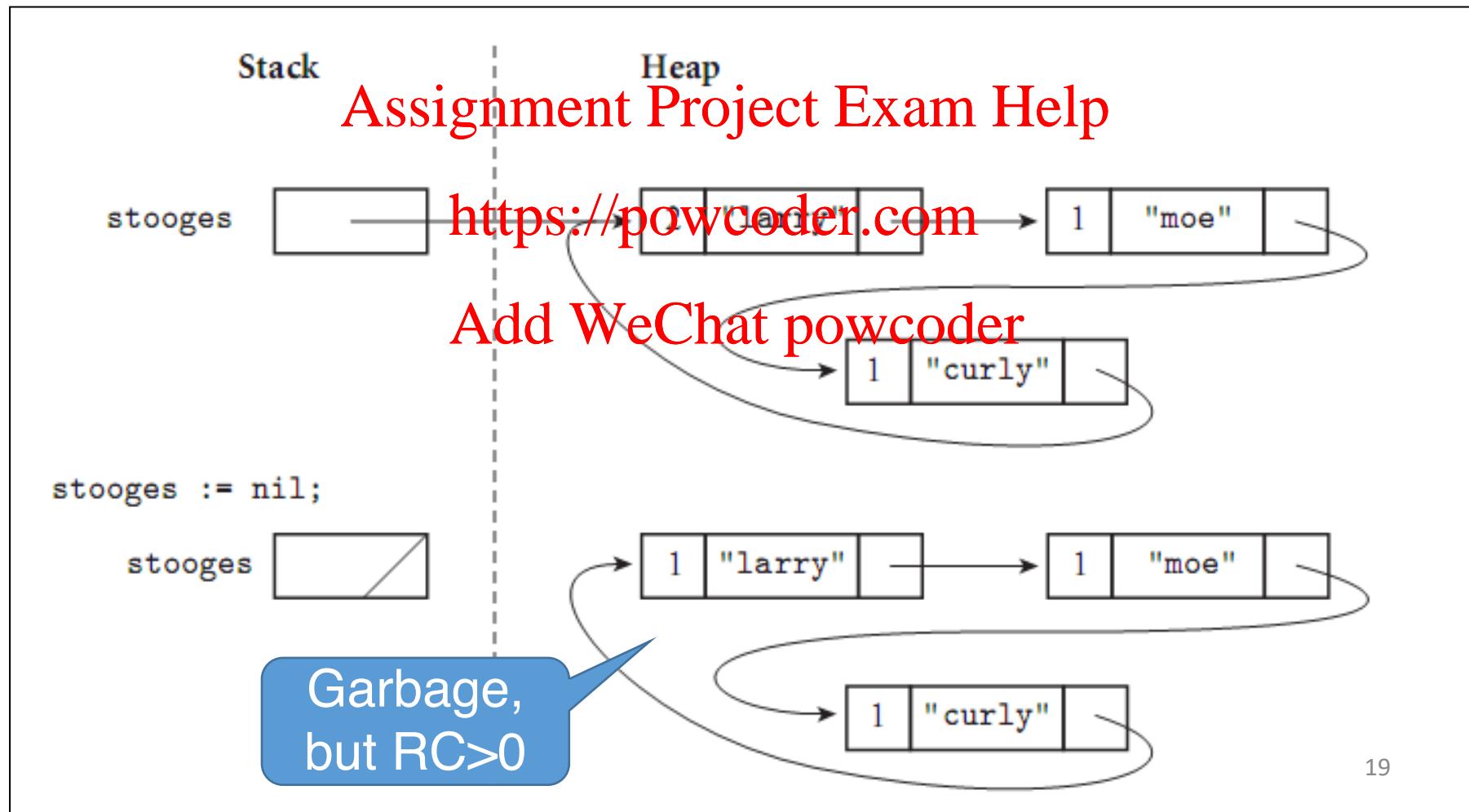
- when one object creates a reference to another object, it owns that object (retain)
- when the object deletes that reference, it relinquishes ownership (release)

Multiple owners of an object

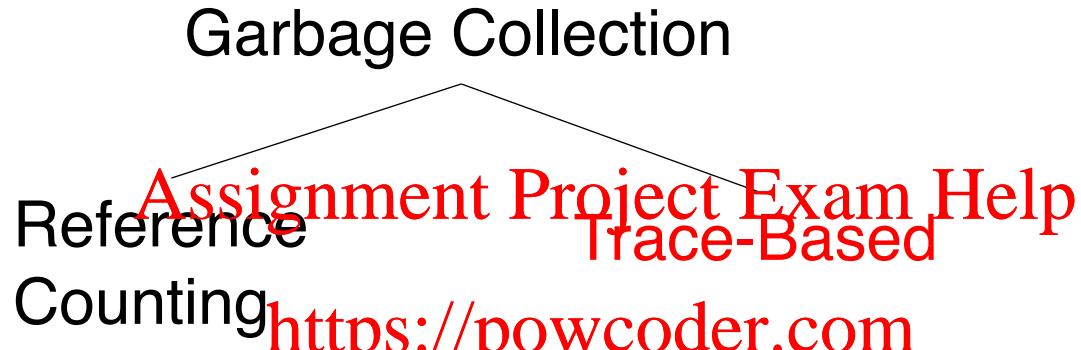
Zero owners of an object

Problem of Reference Counting

Ownership \neq Accessibility



Taxonomy



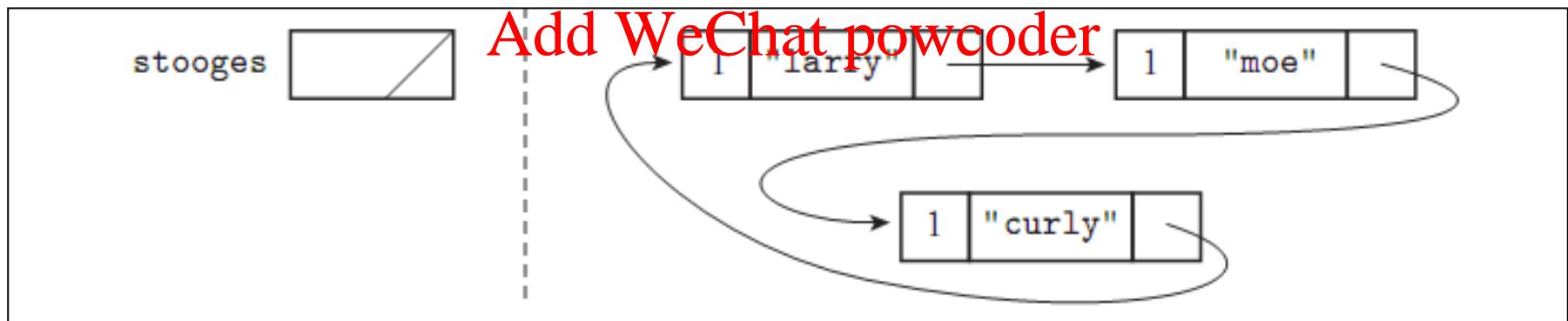
Add WeChat powcoder

What Is Garbage

Ideally, any heap block not used in the future

Assignment Project Exam Help

In practice, the garbage collector identifies blocks
inaccessible from program



Essentially a reachability problem (from alive variables)

GC II: Tracing Collection

What allocated blocks are still accessible (live)?

- Reference in registers, static area, or stack
- Reference from reachable objects to other objects

<https://powcoder.com>

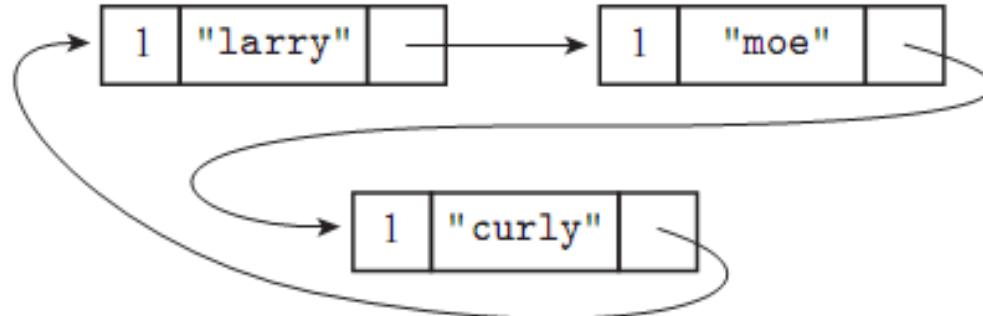
Essentially a reachability problem (heap as directed graph)

Stack

stooges



Heap



#39

Assignment Project Exam Help
OOP and Garbage Collection
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

HW7

- Last assignment
- Due on the last day of class (Dec. 11) midnight
(no late submission)

<https://powcoder.com>

Add WeChat powcoder

Abstract Data Types

Primitive types: values and operations on values

User-defined types: records, lists

[Assignment](#) [Project](#) [Exam](#) [Help](#)

Focus on values

<https://powcoder.com>

ADT: defined by a set of operations on a type

[Add WeChat powcoder](#)

Focus on operation

Stack is a type with new, pop, push, empty ...

Internal representation is less relevant

Classifying Operations

Creators: create new objects of type

Assignment Project Exam Help

Producers: create new objects from old ones
<https://powcoder.com>

Mutators: change objects, e.g., `list.add(n)`

Observers: take objects of ADT and return
objects with different type, e.g., `list.size()`

ADT Example

int

Assignment Project Exam Help

Creators: numeric literals 1, 2, 3, ...
<https://powcoder.com>

Producers: arithmetic operations +, -, *, /, ...
Add WeChat powcoder

Observers: comparison operators ==, !=, <, >

Mutators: none (immutable)

ADT Example

List

Assignment Project Exam Help

Creators: ArrayList, LinkedList, ...
<https://powcoder.com>

Producers: Collections.unmodifiableList()

Observers: size(), get()

Mutators: add(), remove(), ...

ADT Example

String

Assignment Project Exam Help

Creators: String(), String(char[])
<https://powcoder.com>

Producers: concat(), substring(), ...

Observers: length(), charAt(), ...

Mutators: none (immutable)

OOP Terminology

Class: *a richer version of ADT*

Object (Instance): ~~Assignment Project Exam Help~~ a variable of a class (a value of a type)

Field: variable in a class
<https://powcoder.com>

Method: operation in a class
~~Add WeChat powcoder~~

Object-Oriented Programming

Key elements:

- Encapsulation
- Subtyping
- Inheritance

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Encapsulation (Information Hiding)

- Group data and operations in one place (typically, in one class)
- Hide irrelevant details (using visibility modifiers, such as *public*, *private*, *protected*)

Add WeChat powcoder

Subtyping

Sometimes, every value of type B is of type A

- e.g., a `Rectangle` is always a `Shape`

<https://powcoder.com>

We say B is a *subtype* of A, meaning

"every object that satisfies interface of B also satisfies interface of A"

Subtyping

```
interface Shape {  
    public double area();  
    public int edges();  
}
```

Assignment Project Exam Help

<https://powcoder.com>

```
class Circle implements Shape {  
    double radius;  
    public double area() { return 3.14*radius*radius; }  
    public int edges() { return 1; }  
}
```

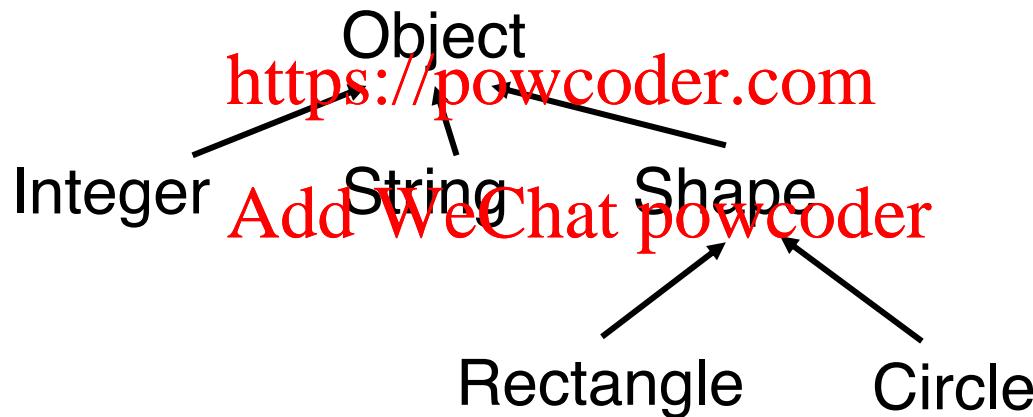
Add WeChat powcoder

Subtyping Relation

We write $A \leq B$ to mean A is a subtype of B

Relation \leq defines a partial ordering on types

Assignment Project Exam Help



Reflexivity: $\forall T. T \leq T$

Antisymmetry: $\forall T_1, T_2 . T_1 \leq T_2 \wedge T_2 \leq T_1 \Rightarrow T_1 = T_2$

Transitivity: $\forall T_1, T_2, T_3 . T_1 \leq T_2 \wedge T_2 \leq T_3 \Rightarrow T_1 \leq T_3$

Subtype Polymorphism

“Casting” a subtype to its supertype is ALWAYS safe (known as “Upcast”)

Assignment Project Exam Help

Hence, a function with parameter of type B can take any data with type $A \leq B$

Add WeChat [powcoder](https://powcoder.com)

```
int f (Object o) {...} // takes object of any type
int g (Shape s) {...} // takes object of subtype
                      // of Shape, such as Circle
```

Upcasting and Downcasting

- Upcast: change to a supertype
 - Always safe
- Downcast: change to a subtype
 - Not always safe

```
Rectangle r = new Rectangle(2,3); // OK  
Shape s = r; // Upcast is always safe  
r = (Rectangle) s; // OK, but downcast  
// is not always safe
```

Java checks type casts at run time

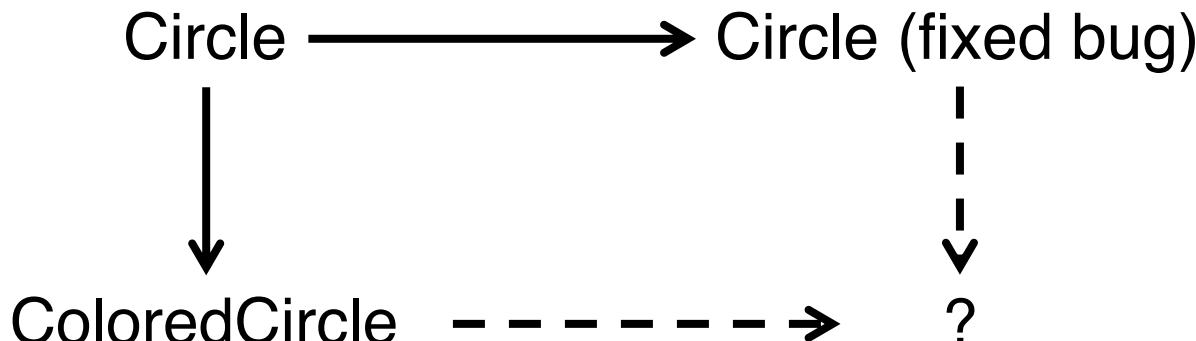
Motivation of Inheritance

```
class Circle implements Shape {  
    double radius;  
    public double area() {return 3.14*radius*radius};  
    public int sides() {return 1};  
}
```

<https://powcoder.com>

How to implement ColoredCircle?

Option 1: copy-and-paste
[Add WeChat powcoder](#)



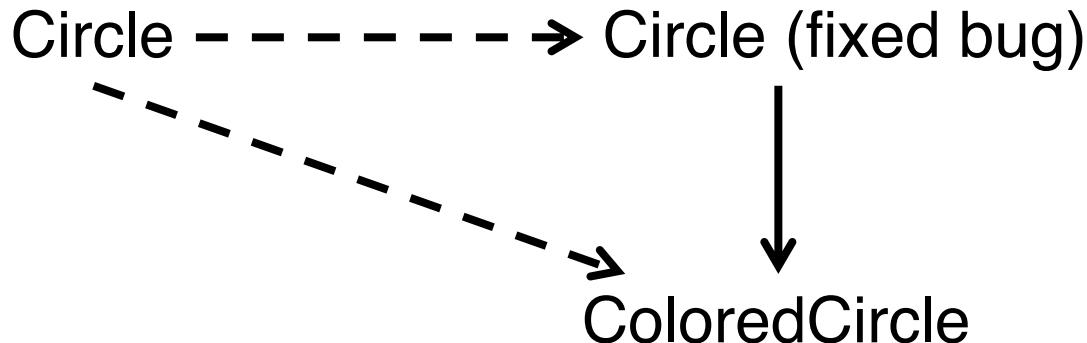
Inheritance

```
class ColoredCircle extends Circle {  
    private Color color;  
    ...  
    Color getColor {return color;}  
    // methods area, edges are inherited from Circle  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Inheritance

```
class ColoredCircle extends Circle {  
    private Color color;  
    ...  
    Color getColor {return color;}  
    // methods area, edges are inherited from Circle  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Inheritance introduces subtyping:

ColoredCircle **inherits all fields & methods**

Circle **is the supertype of** ColoredCircle

ColoredCircle **is the subtype of** Circle

Overriding

```
class DoubleCircle extends Circle {  
    public DoubleCircle(double r) {super(r);}  
    public int areas() {return 2;  
}
```

<https://powcoder.com>

Add WeChat powcoder
Subclass may redefine methods in super class

Object-Oriented Programming

Key elements:

- Encapsulation [Assignment](#) [Project](#) [Exam](#) [Help](#)
- Subtyping <https://powcoder.com>
- Inheritance [Add WeChat powcoder](#)

How are these features implemented?

Running Example

Assignment Project Exam Help
Circle: edges return 1

subtype of

<https://powcoder.com>

subtype of

ColoredCircle: inherits edges
has a new filed “color”

Add WeChat powcoder

DoubleCircle: edges return 2

Memory Layout (Fields)

An object has

- Fields (and ones from super class)
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Circle object1:

DoubleCircle object:

ColoredCircle object:

<https://powcoder.com>

Add WeChat powcoder

radius

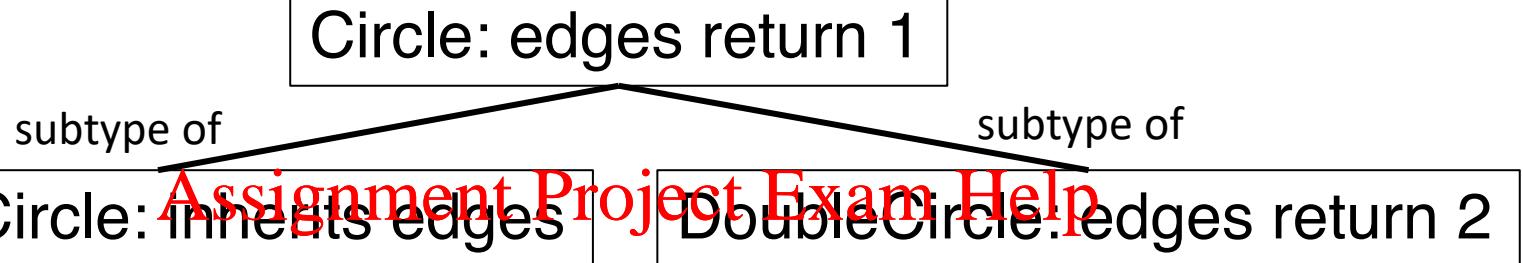
radius

radius
color

```
foo (Circle s) {  
    s.radius; // offset?  
}
```

Caution: new fields in subtype should extend inherited fields due to subtype polymorphism

Memory Layout (Functions)



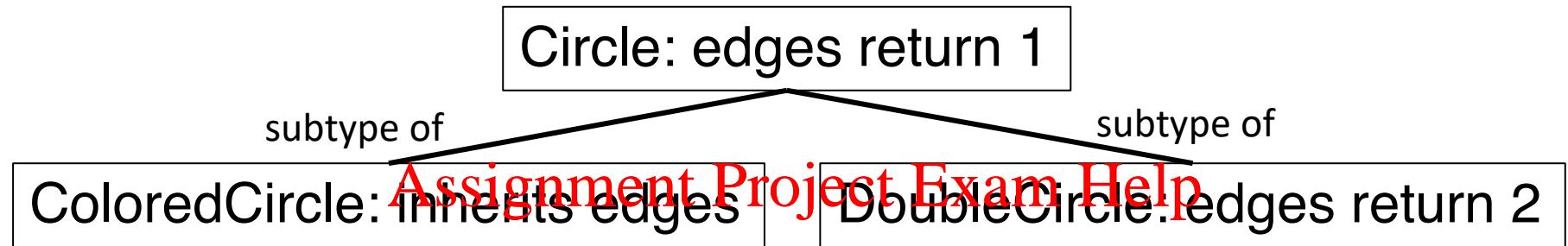
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      // Which implementation?  
}
```

Static dispatch: s.edges() always returns 1

Dynamic dispatch: return value of s.edges() controlled by the type of s

Static Dispatch



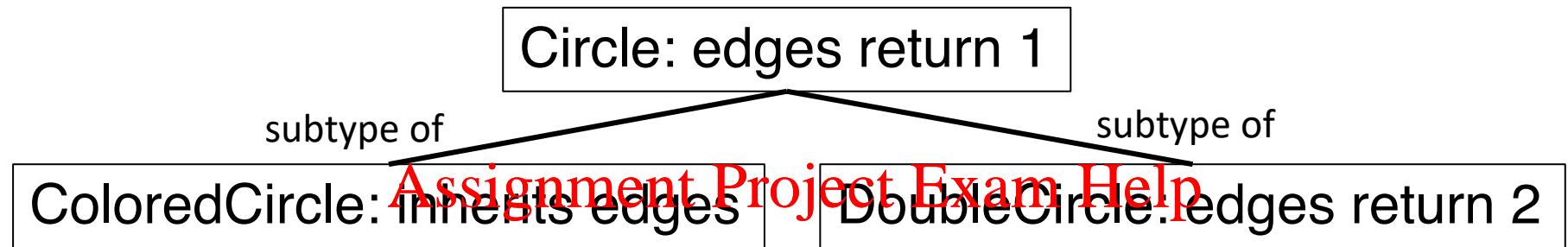
<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      /Add WeChat powcoder  
}
```

Dispatch to the implementation in class Circle

Hence, `s.edges()` always returns 1

Implementation: Static



<https://powcoder.com>

```
foo (Circle s) {  
    s.edges();      /Add WeChat powcoder  
}
```

The compiler can always tell which implementation at compile time (e.g., the `edges` method in class `Circle`)

#40

Assignment Project Exam Help
OOP and Garbage Collection
<https://powcoder.com>

Add WeChat powcoder

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2020

Stop-the-World Collectors

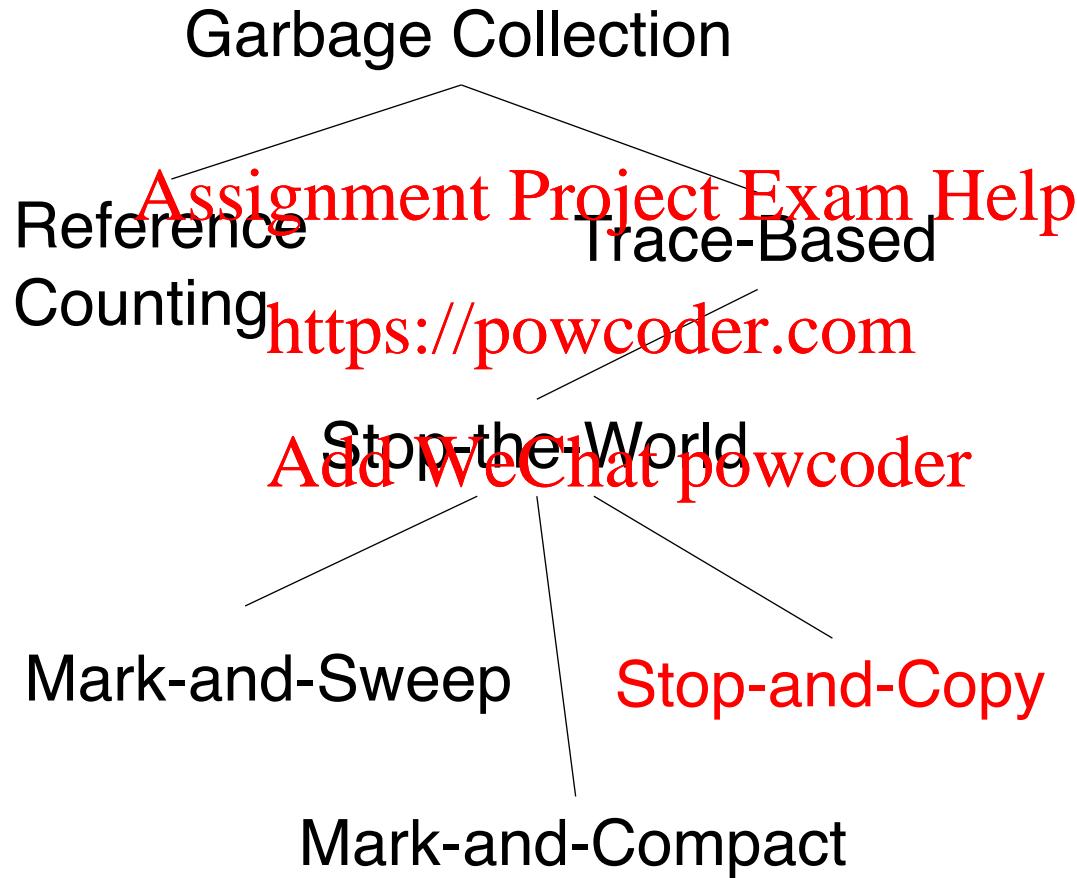
A: number of alive objects; N: all objects on heap

H: total heap size

Assignment Project Exam Help

	Mark-And-Sweep	Mark-And-Compact
Complexity	$O(N)$	$O(N)$
Fragmentation?	Yes	No
Memory move	No	Yes
Effective heap size	H	H

Taxonomy



Stop-and-Copy (Copying Collector)

Time-Space Tradeoff:

- 2 Heaps: allocate space in one, copy to second when the other is full (half of the heap is unused)
- Only one pass <https://powcoder.com> is needed (much faster than previous algorithms)

Add WeChat powcoder

MB bit is not needed

Stop-and-Copy

Triggered when heap in use is full, interrupt program

For each visited object

[Assignment](#) [Project](#) [Exam](#) [Help](#)

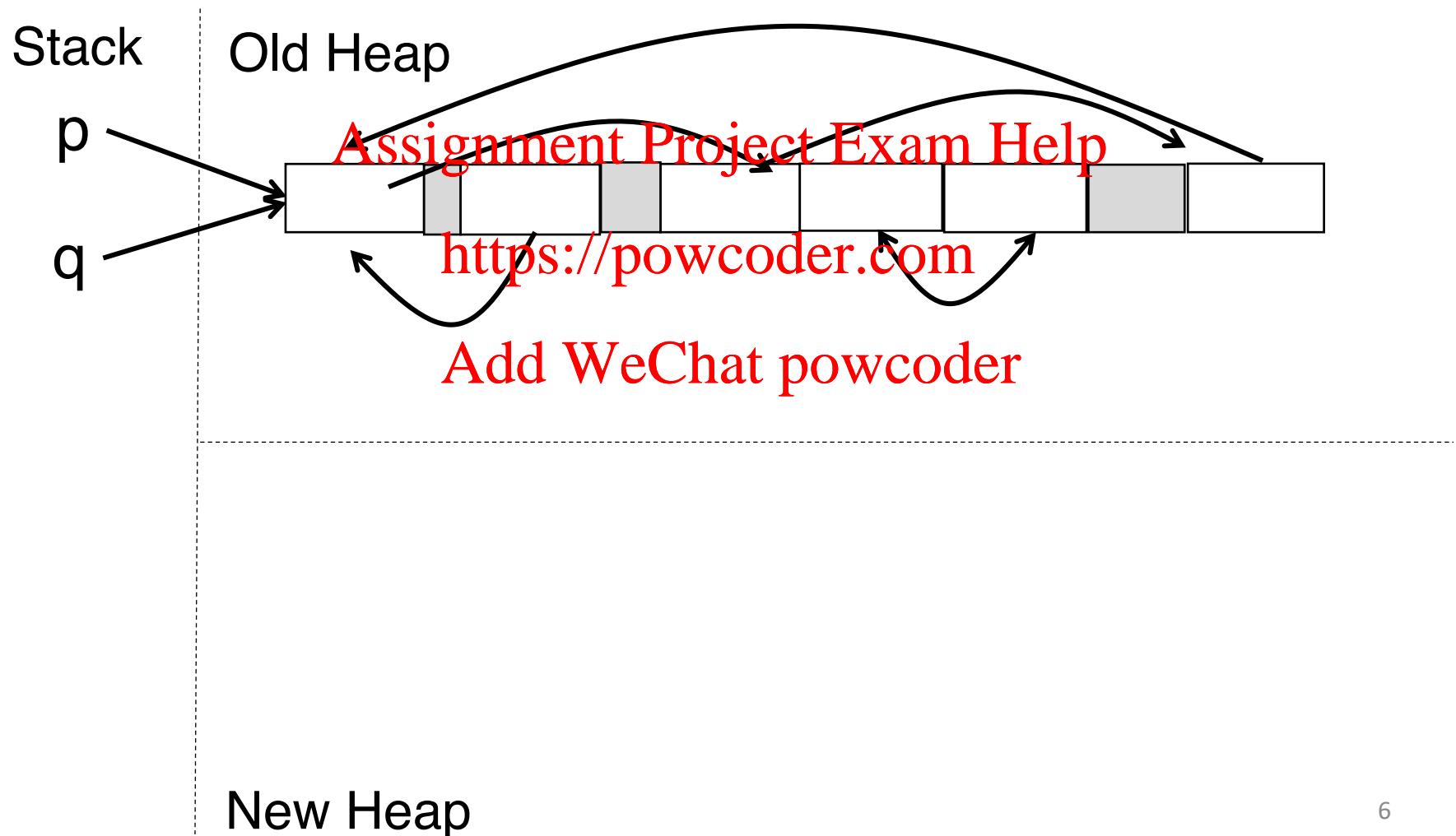
1. Copy it to the other heap (no fragmentation)
2. For the object in the old heap, keep a *forwarding pointer* to the new address
3. For each object it points to:
 - a) if visited: update links (follow forwarding pointer)
 - b) otherwise, (recursively) visit object and update link

Retarget root references

Switch heaps

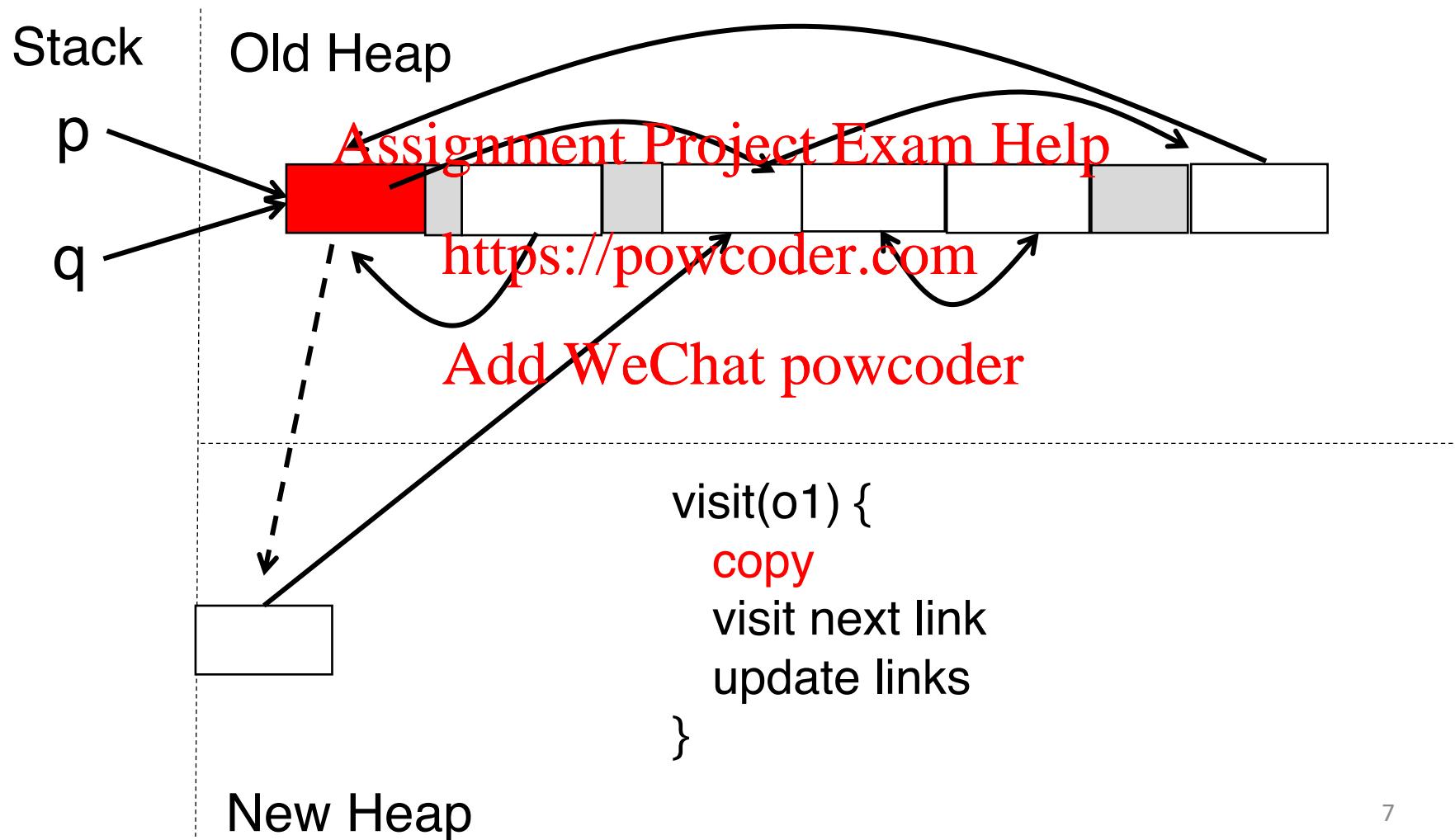
Example

Initial state



Example

Visit red node

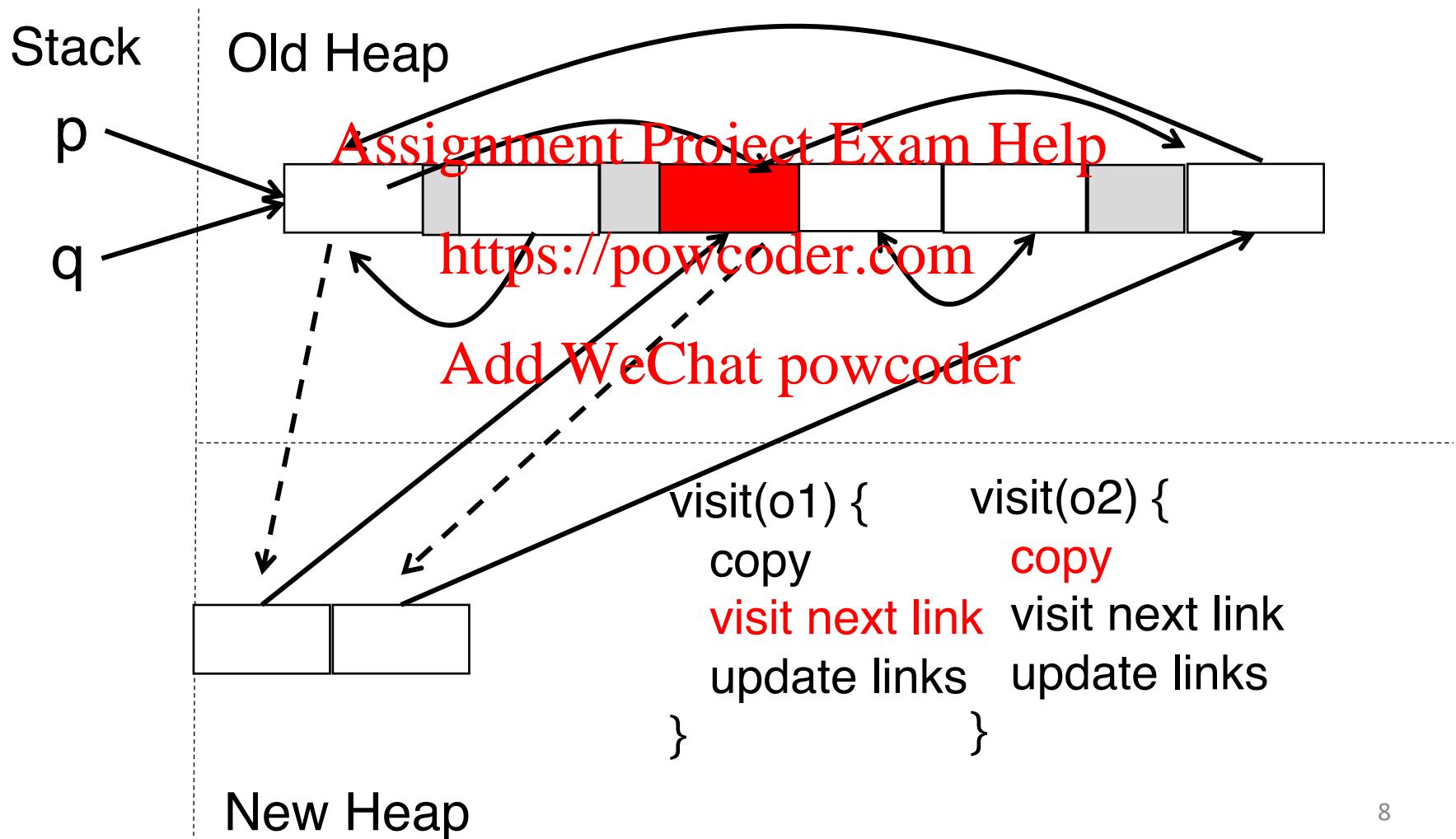


Example

Forwarding
Pointer

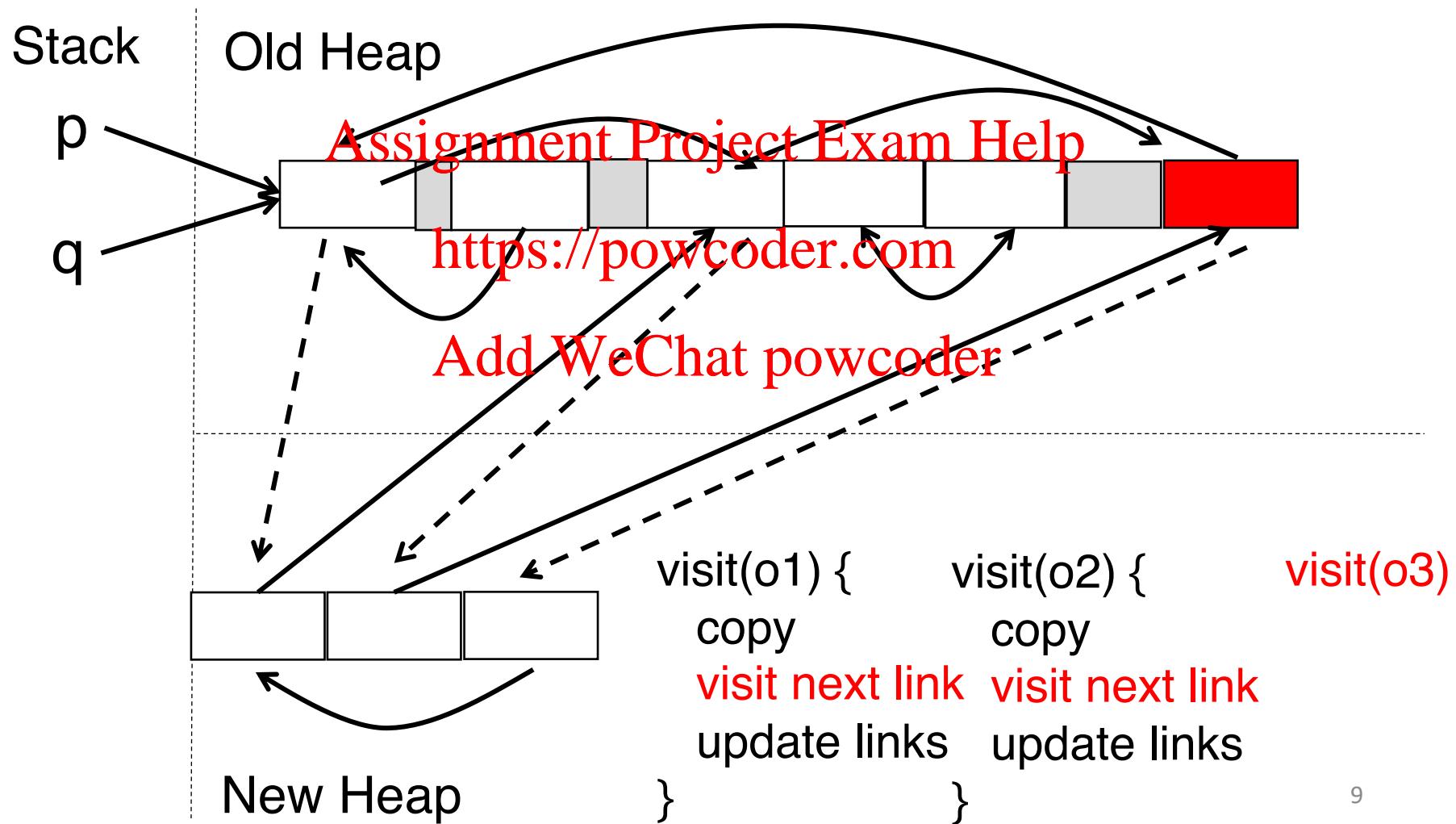


Visit red node (following link from new heap)



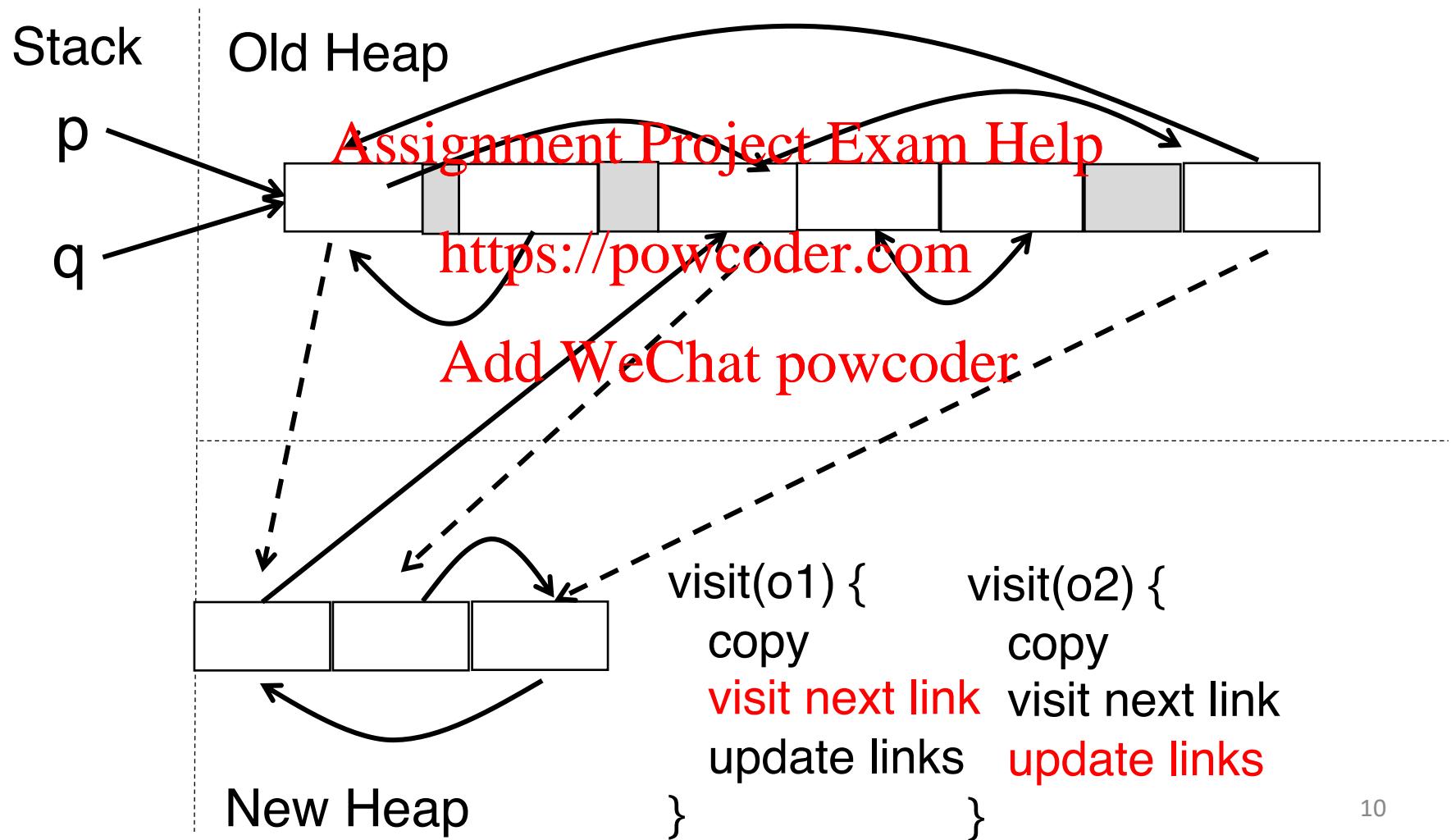
Example

Copy red node (internal link is updated)



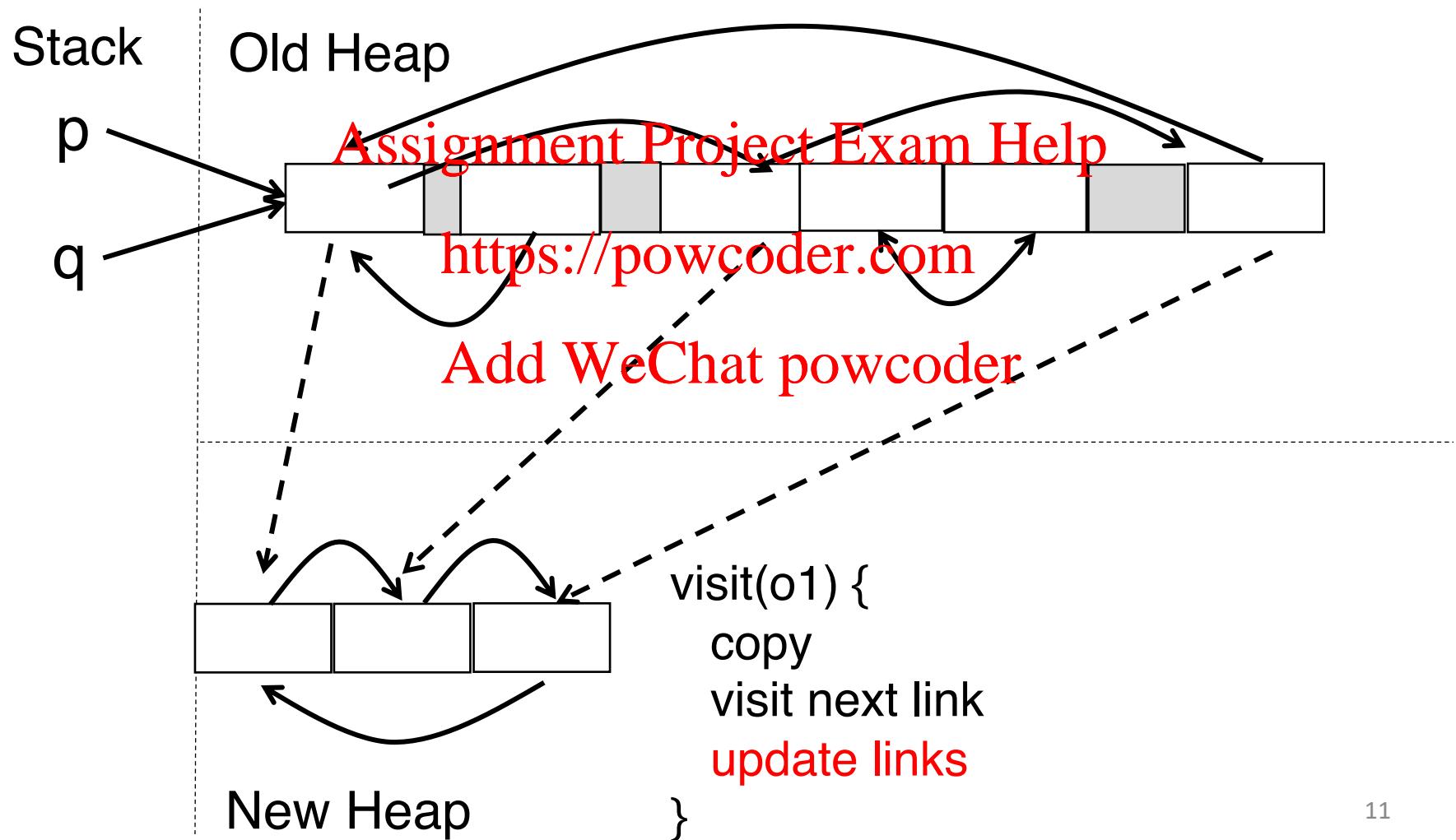
Example

Update links



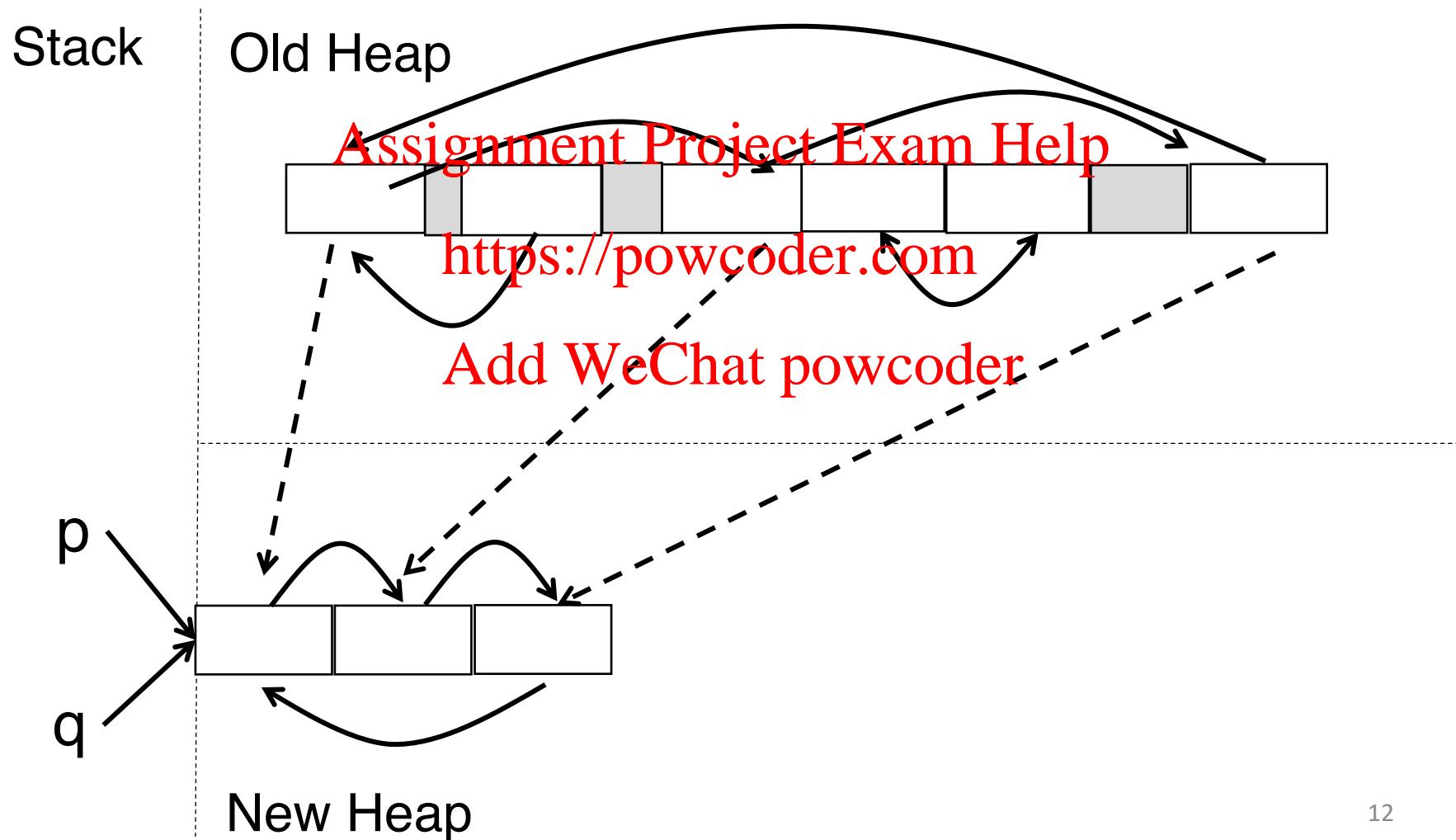
Example

Update links



Example

Retarget root references



Example

Swap Heaps (no need to clean the old one)

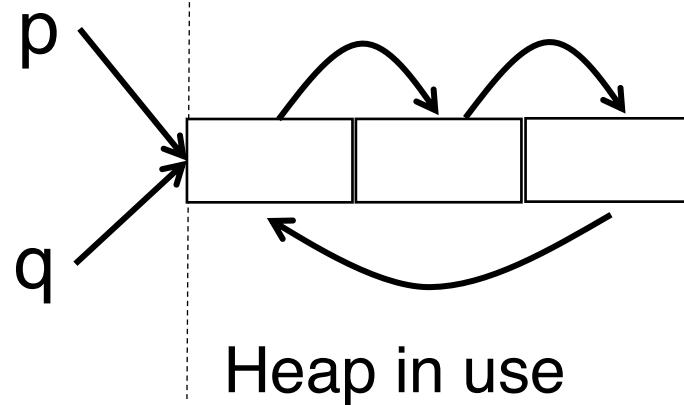
Stack

Heap for copying

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Stop-the-World Collectors

A: number of alive objects; N: all objects on heap

H: total heap size

Assignment Project Exam Help

	Mark-And-Sweep	Mark-And-Compact	Stop-And-Copy
Complexity	$O(N)$	$O(N)$	$O(A)$
Fragmentation?	Yes	No	No
Memory move	No	Yes	Yes
Effective heap size	H	H	$H/2$

Stop-and-Copy

Pros:

Eliminate fragmentation

Assignment Project Exam Help

Collection time proportional to *live* objects

Great for short-lived data

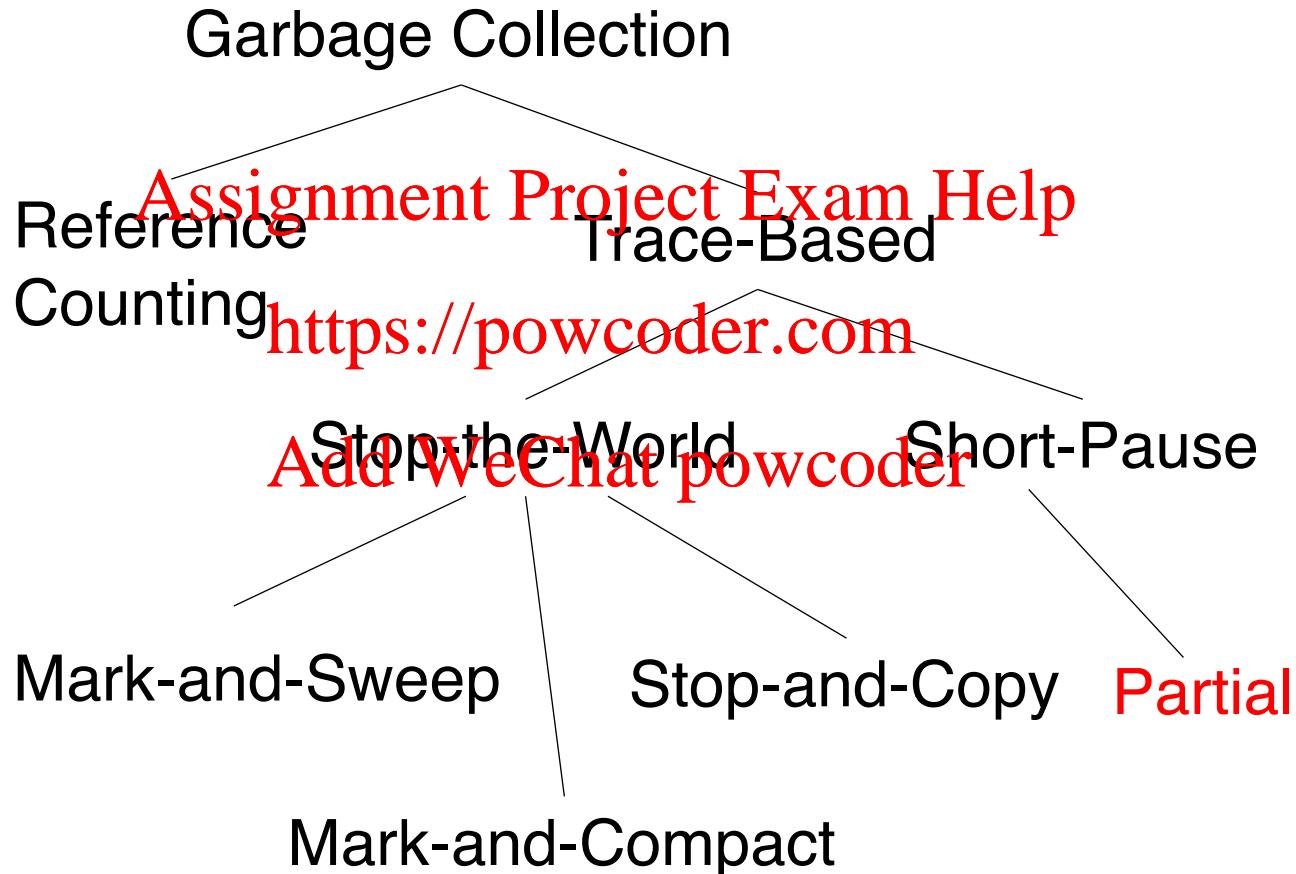
Add WeChat powcoder

Cons:

Only half of heap is in use, requires more collecting
(virtual memory alleviates this limitation)

Bad for long living data (copying data back and forth)

Taxonomy



Short-Pause Collection

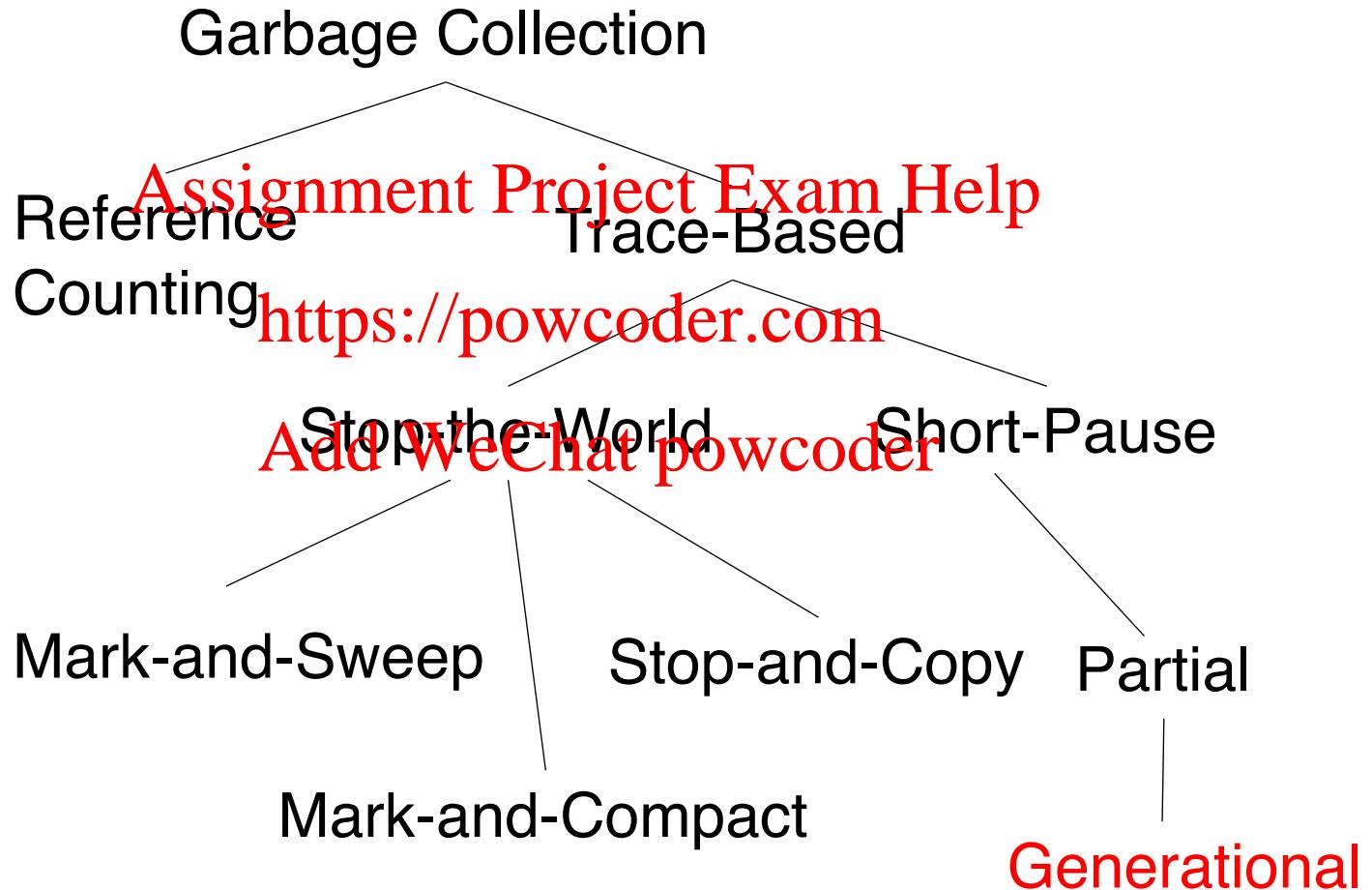
Partial

- stop the program, but only briefly, to garbage collect **a part of** the heap

[Assignment Project Exam Help
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

Taxonomy



The Object Life-Cycle

“Most objects die young.”

- But those that survive one GC are likely to survive many

Assignment Project Exam Help

<https://powcoder.com>

Idea: tailor GC to spend more time on regions of
the heap where objects have just been created

Add WeChat powcoder

A better ratio of reclaimed space per unit time

Generational

Divide heap into generations $g_1, g_2 \dots$

- g_i holds older objects than g_{i+1}

[Assignment Project Exam Help](https://powcoder.com)
Create new objects in g_1 , until it fills up

Add WeChat powcoder

GC g_1 only; move reachable objects to g_2 after several collections (typically one collection)

Generational

When g_2 fills, garbage collect g_1 and g_2 , and put the reachable objects in g_3

[Assignment](#) [Project](#) [Exam](#) [Help](#)

In general: When g_i fills, collect g_1, g_2, \dots, g_i , and put the reachable objects in g_{i+1}

[Add WeChat](#) [powcoder](#)

**What GC algorithm is better for young generation?
How about old generation?**

Algorithm for Young Generation

A: number of alive objects; N: all objects on heap

A << N Assignment Project Exam Help

	Mark-And-Sweep	Mark-And-Compact	Stop-And-Copy
Complexity	$O(N)$	$O(N)$	$O(A)$
Fragmentation?	Yes	No	No
Memory move	No	Yes	Yes
Effective heap size	H	H	$H/2$

Stop-and-copy is the most efficient

Algorithm for Old Generation

A: number of alive objects; N: all objects on heap

A is close to N
Assignment Project Exam Help

	Mark-And-Sweep	Mark-And-Compact	Stop-And-Copy
Complexity	$O(N)$	$O(N)$	$O(A)$
Fragmentation?	Yes	No	No
Memory move	No	Yes	Yes
Effective heap size	H	H	$H/2$

Mark-And-Compact removes fragmentation

Algorithm for Old Generation

A: number of alive objects; N: all objects on heap

What if most objects have the same size?

	Mark-And-Sweep https://powcoder.com	Mark-And-Compact	Stop-And-Copy
Complexity	$O(N)$ Add WeChat powcoder	$O(N)$	$O(A)$
Fragmentation?	Yes	No	No
Memory move	No	Yes	Yes
Effective heap size	H	H	$H/2$

Mark-And-Sweep saves the cost of moving objects

Generational

Pros:

Divide heap according to lifetimes of data
[Assignment Project Exam Help](#)

Great for data with mixed lifetimes

<https://powcoder.com>

Cons:

[Add WeChat powcoder](#)

Garbage might survive a collection

Small heap size for each generation

More frequent collection

Case Study: Java SE

Generational Collectors

- Young Generation, 3 partitions:
 - Eden, 2 Survivor Spaces (**stop-and-copy**)
 - Minor collections ([high mortality rate](https://powcoder.com))
- Old Generation
 - surviving objects promoted from Young Generation
 - Major collections (infrequent)
- Permanent Generation
 - objects needed by JVM, Strings

CMPSC 461: Programming Language Concepts

Final Exam Practice Questions

Use these problems **in addition** to previous HWs, midterms, practice problems to prepare for the final exam.

Problem 1 There are two approaches of checking the validity of a Hoare triple: forward reasoning and backward reasoning. The former starts from the desired precondition, and checks whether the computed strongest postcondition for the program implies the desired postcondition; the latter starts from the desired postcondition, and computes the weakest precondition rather than the strongest postcondition. What makes the second approach more appealing and popular in practice?

Problem 2 Use the following three rules we gave in lecture to prove that the following two Hoare triples are valid.

- $\text{wp}(x := e, Q) = Q[x \leftarrow e]$
- $\text{wp}(s_1; s_2, Q) = \text{wp}(s_1, \text{wp}(s_2, Q))$
- $\text{wp}(\text{if } (E) s_1 \text{ else } s_2, Q) = (E \Rightarrow \text{wp}(s_1, Q)) \wedge (\neg E \Rightarrow \text{wp}(s_2, Q))$

1. $\{x > -1\}$
 $y := x + x;$
 $y := y + 2;$
 $\{y > 0\}$

2. $\{y \geq 0\}$
if ($y < 1$) $x := y + 1;$
else $x := y;$
 $\{x > 0\}$

Assignment Project Exam Help

<https://powcoder.com>

Problem 3 The following program sets the return value r to $1 + 2 + \dots + n$, where n is an input. Write down a loop invariant and show that the Hoare triple is valid. Recall that a loop invariant should be 1) true before the first execution of the loop, 2) true after the execution of each loop iteration, given that the loop condition and the invariant are both true before the iteration, and 3) strong enough to prove the correctness of the code (i.e., $r = \text{Sum}(n)$ is true after the loop, if it terminates).

To simplify your notation, assume $\text{Sum}(n)$ is a function in logics that returns $\sum_{i=1}^n i$.

```
{n>0}
i := 1;
r := 0;
while (i ≤ n) {
    r := r + i;
    i++;
}
{r = Sum(n)}
```

Problem 4 What are the key elements in any OOP language? Briefly explain every element in your own words.

Problem 5 What is the difference between static and dynamic dispatching? Which dispatching method is used to implement subtype polymorphism?

Problem 6 Consider the following classes defined in Java, which uses dynamic dispatching.

```

class A {
    public int a () { return 5; }
    public void b () { System.out.println( "A" + a()); }
}

class B extends A {
    public void b () { System.out.println( "C" + a()); }
}

class C extends B {
    public int a() {return 10; }
}

```

Given a polymorphic function `void func(A a) {a.b();}`, what are the outputs of the following code?

```

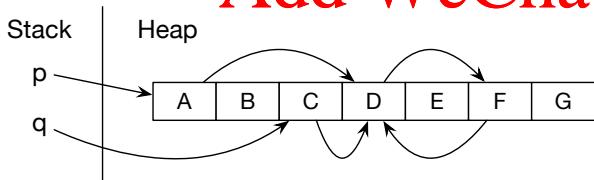
B b = new B();
C c = new C();
func(b);
func(c);

```

Problem 7 Assignment Project Exam Help

Consider the heap state as shown below before garbage collection. For each of the following algorithms, write down the objects being visited during the collection in sequence. Assume 1) reachability analysis uses depth-first search, which will skip objects that have already been visited, 2) search starts from p, and then q, 3) when the entire heap is traversed, objects are visited from left to right. You don't need to write down the newly created object copies, if any.

Add WeChat powcoder



1. Mark-and-Sweep
2. Mark-and-Compact
3. Stop-and-Copy

Problem 3 The following program sets the return value r to $1 + 2 + \dots + n$ where n is an input. Write down a loop invariant and show that the Hoare triple is valid. Recall that a loop invariant should be 1) true before the first execution of the loop, 2) true after the execution of each loop iteration, given that the loop condition and the invariant are both true before the iteration, and 3) strong enough to prove the correctness of the code (i.e., $r = \text{Sum}(n)$ is true after the loop, if it terminates).

To simplify your notation, assume $\text{Sum}(n)$ is a function in logics that returns $\sum_{i=1}^n i$.

```
(n > 0)
i := 1;
r := 0;
while (i ≤ n) {
    r := r + i;
    i++;
}
{r = Sum(n)}
```

loop invariant:

$$\text{Inv} \triangleq r = \text{Sum}(i-1) \wedge i \leq n+1$$

$$n = 3$$

before loop

Assignment Project Exam Help

① $\{n > 0\}$

$$n: 3, i: 1, r: 1 \text{ sum}(1)$$

$$i := 1$$

$v := 0$ is valid

after one iteration:

$$n: 3, i: 2, r: 3 \text{ sum}(2)$$

{ Inv }

after three iteration:

Add WeChat powcoder

wp($i := 1; r := 0, \text{Inv}$)

$$= wp(i := 1, wp(r := 0, \text{Inv}))$$

$$= wp(i := 1, 0 = \text{Sum}(i-1) \wedge i \leq n+1)$$

$$= 0 + \text{Sum}(0) \wedge 1 \leq n+1$$

$$= 0 \leq n$$

$n > 0 \Rightarrow 0 \leq n$ so triple is valid

$n=3$

before loop:

$n=3, i=1, r=0$

after one iteration: $i \leq n \Rightarrow 1 \leq 3 \checkmark$

$n=3, i=2, r=1$
 $i++ \quad r=r+i$

sum(i)

after two iteration: $2 \leq 3 \checkmark$

$n=3, i=3, r=3$

sum(j)

after three iteration: $3 \leq 3 \checkmark$

$n=3, i=4, r=6$

sum(i)

Assignment Project Exam Help

loop invariant:
 $\forall i \leq r \quad r = \text{sum}(\underline{i-1}) \wedge i \leq n+1$

③ Invariant: $r = \text{sum}(n)$
<https://powcoder.com>

Add WeChat powcoder

```

class A {
    public int a () { return 5; }
    public void b () { System.out.println( "A" + a()); }
}

class B extends A {
    public void b () { System.out.println( "B" + a()); }
}

class C extends B {
    public int a() {return 10;}
}

```

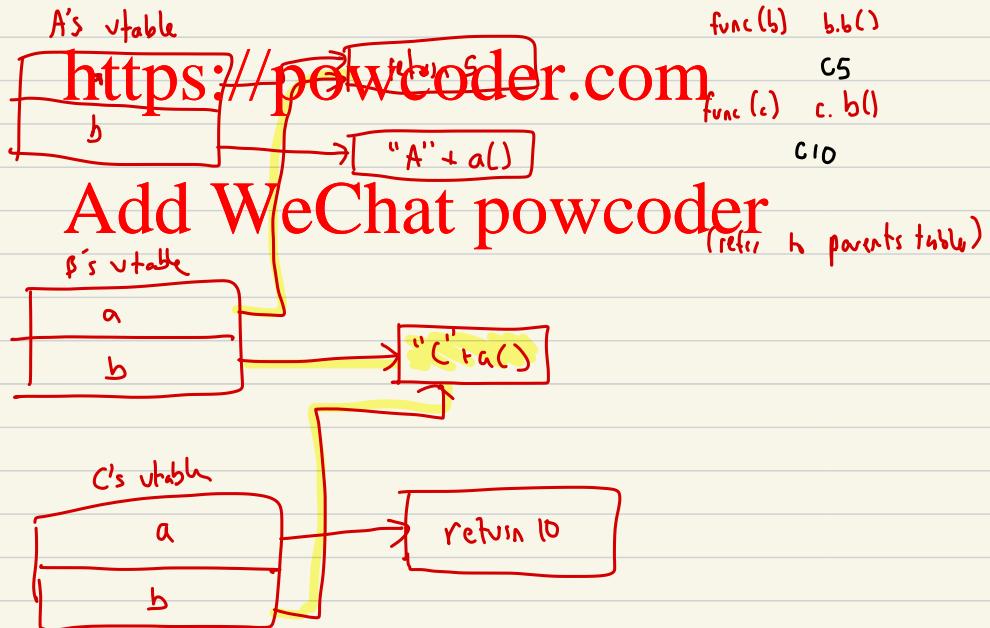
Given a polymorphic function `void func(A a){a.b();}`, what are the outputs of the following code?

```

B b = new B();
C c = new C();
func(b);
func(c);

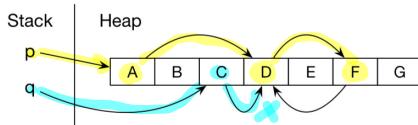
```

Assignment Project Exam Help



Problem 7

Consider the heap state as shown below before garbage collection. For each of the following algorithms, write down the objects being visited during the collection in sequence. Assume 1) reachability analysis uses depth-first search, which will skip objects that have already been visited, 2) search starts from p, and then q, 3) when the entire heap is traversed, objects are visited from left to right. You don't need to write down the newly created object copies, if any.



1. Mark-and-Sweep (DFS) *depth-first-search*
2. Mark-and-Compact

Assignment Project Exam Help

1) *Mark : A, D, F, G*

https://powcoder.com

2) *Mark : A, D, F, C*

Add WeChat powcoder

3) *Skip & copy : ADFC*