https://powcoder.com

Assignment Project Exam Help

Add WeChat powcoder

Control in Sequential Languages

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Mitchell Chapter 8

https://powcoder.com

Topics

Assignment Project Exam Help

Add WeChat powcoder

- Exceptions
  - "structured" jumps that may return a value
  - dynamic scoping of exception handler
- Continuations
  - Function representing the rest of the program
  - Generalized form of tail recursion
- Control of evaluation order (force and delay)
  - Can increase efficiency
  - *Call-by-need* parameter passing.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Exceptions: Structured Exit

Assignment Project Exam Help

Add WeChat powcoder

- Historically, `goto` statements were used, which can jump out of anywhere or into anywhere
- Some languages have `break` statements
- Exceptions provide a *clean* way to jump out of or abort a function call.
  - Their effects would not be easy to achieve with other forms of controlled jumps.
- Main language constructs:
  - Statement or expression to *raise* or *throw* exception
  - Statement or expression to *handle* or *catch* exceptions, called a *handler*

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Exceptions: Structured Exit

Assignment Project Exam Help

Add WeChat powcoder

Terminate part of computation, achieving the following effects:

- Jump out of construct
- Pass data as part of jump
  - This data can be used, for example, to recover from an error.

Assignment Project Exam Help

- Return to most recent site set up to handle exception
  - The correct handler is determined according to dynamic scoping rules

https://powcoder.com

Add WeChat powcoder

- Unnecessary activation records may be deallocated
  - May need to free heap space, other resources

https://powcoder.com

Assignment Project Exam Help

C++ vs ML Exceptions

Add WeChat powcoder

- C++ exceptions
  - Can throw any type
  - Stroustrup: "I prefer to define types with no other purpose than exception handling. This minimizes confusion about their purpose. In particular, I never use a built-in type, such as int, as an exception." - The C++ Programming Language, 3rd ed.

Assignment Project Exam Help

- ML exceptions
  - Exceptions are a different kind of entity than types.
  - Declare exceptions before use

https://powcoder.com

Add WeChat powcoder

Similar, but ML requires the recommended C++ style.

https://powcoder.com
Assignment Project Exam Help

OCaml Exceptions

Add WeChat powcoder

- Declaration

  exception ⟨name⟩ of ⟨type⟩

  – gives name of exception and type of data passed when raised

- Raise

  raise (⟨name⟩ ⟨parameters⟩)

  – expression form to raise an exception and pass data

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

- Handler

  try ⟨exp1⟩ with | ⟨pattern⟩ -> ⟨exp2⟩

  – Evaluate first expression.

  – If exception that matches pattern is raised, then evaluate second expression instead.

  – General form allows multiple patterns.

https://powcoder.com

Examples

Assignment Project Exam Help

Add WeChat powcoder

```
exception Ovflw                raise Ovflw
exception Signal of int        raise (Signal (x+4))

let f x = if x<min then raise Ovflw else 1/x
(try f x with | Ovflw -> 0) / (try f 0 with | Ovflw -> 1)

let g x = if x=0 then raise (Signal 0)
          else if x=1 then raise (Signal 1)
          else if x=10 then raise (Signal (x-8))
          else (x-2) mod 4
try g 10 with | Signal 0 -> 0
              | Signal 1 -> 1
              | Signal x -> x+8
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Which Handler is Used?

Assignment Project Exam Help

Add WeChat powcoder

```
let f x = if x<min then raise Ovflw else 1/x
(try f x with | Ovflw -> 0) / (try f 0 with | Ovflw -> 1)
```

- Dynamic scoping of handlers
  - First call handles exception one way
  - Second call handles exception another
  - General dynamic scoping rule

    Jump to most recently established handler on run-time stack
- Dynamic scoping is not an accident
  - User knows how to handler error
  - Author of library function does not

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

General Form of Handler Expressions

Assignment Project Exam Help

Add WeChat powcoder

try <exp> with

| <pattern$_1$> -> <exp$_1$>

| <pattern$_2$> -> <exp$_2$>

…

| <patern$_n$> -> <exp$_n$>

Assignment Project Exam Help

https://powcoder.com

- First, <exp> is evaluated.
- If the evaluation terminates normally, the value of the whole try expression is the value of this expression; the handler is never invoked.

Add WeChat powcoder

- If the evaluation raises an exception that matches <pattern$_i$> (and there is no matching handler declared in <exp>), then the corresponding handler is invoked.
- Pattern matching works just as in ordinary OCaml.

https://powcoder.com

Exception for Error Condition

Assignment Project Exam Help

Add WeChat powcoder

```
type 'a tree = Leaf of 'a | Node of 'a tree * 'a tree
exception No_Subtree
let lsub (t:'a tree) =
  match t with
  | Leaf x -> raise No_Subtree
  | Node (x,y) -> x
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- This function raises an exception when there is no reasonable value to return
- We'll look at typing later.

https://powcoder.com

Exception for Efficiency

Assignment Project Exam Help

Add WeChat powcoder

- Function to multiply values of tree leaves

```
let rec prod (t:int tree) : int =
  match t with
  | Leaf x -> x
  | Node (x,y) -> (prod x) * (prod y)
```

Assignment Project Exam Help

- Optimize using exception

https://powcoder.com

```
let exception Zero in
  let rec prod (t:int tree) : int =
    match t with
    | Leaf x -> if x=0 then (raise Zero) else x
    | Node (x,y) -> (prod x) * (prod y)
  in
    try (prod t) with Zero -> 0
```

Add WeChat powcoder

https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try h 1 with X -> 2
     in
     try g f with X -> 4)
with X -> 6
```

scope

handler

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Which handler is used?

https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- When a handler is in a nested block, the handler expression goes on the stack first and is treated like a declaration.
- A handler in a function definition is treated like a local variable.

https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```

handler X	6

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

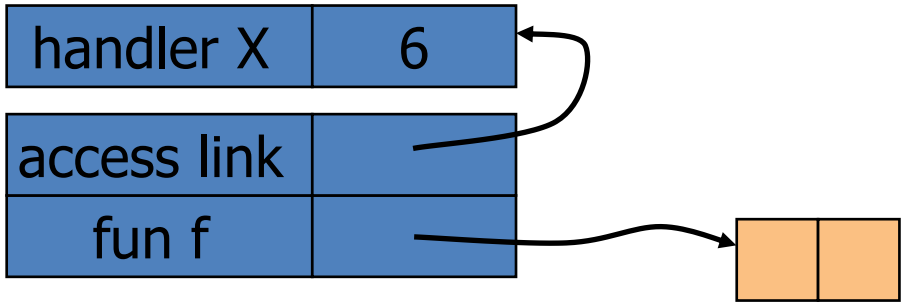
https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Note: pointers in closures left out of diagram, but can be deduced.

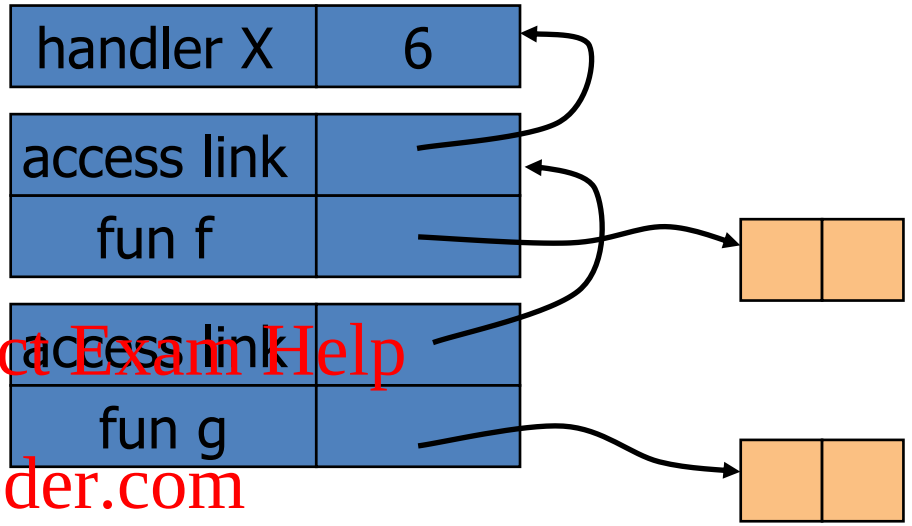
https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Note: pointers in closures left out of diagram, but can be deduced.

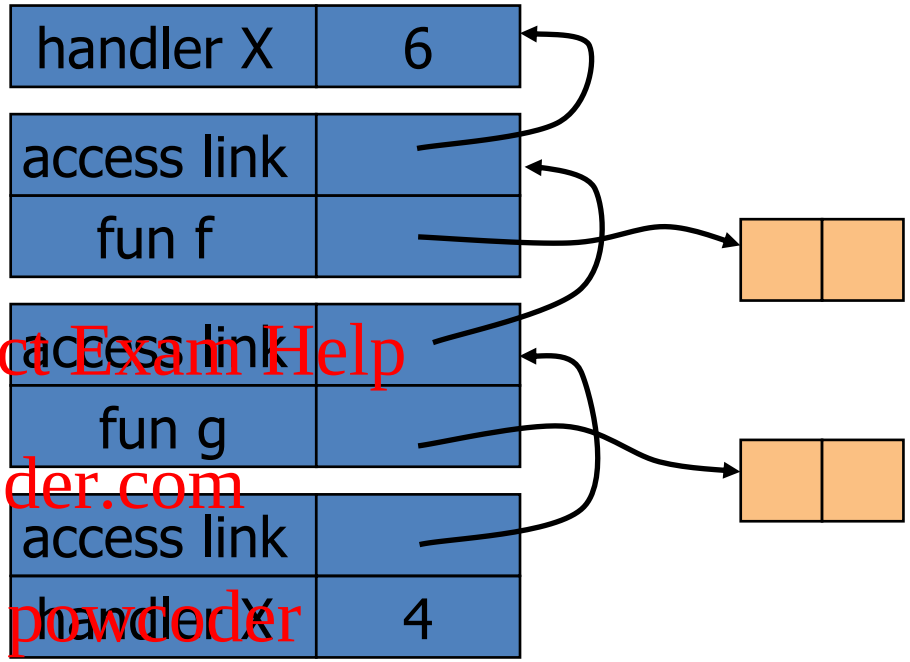
https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

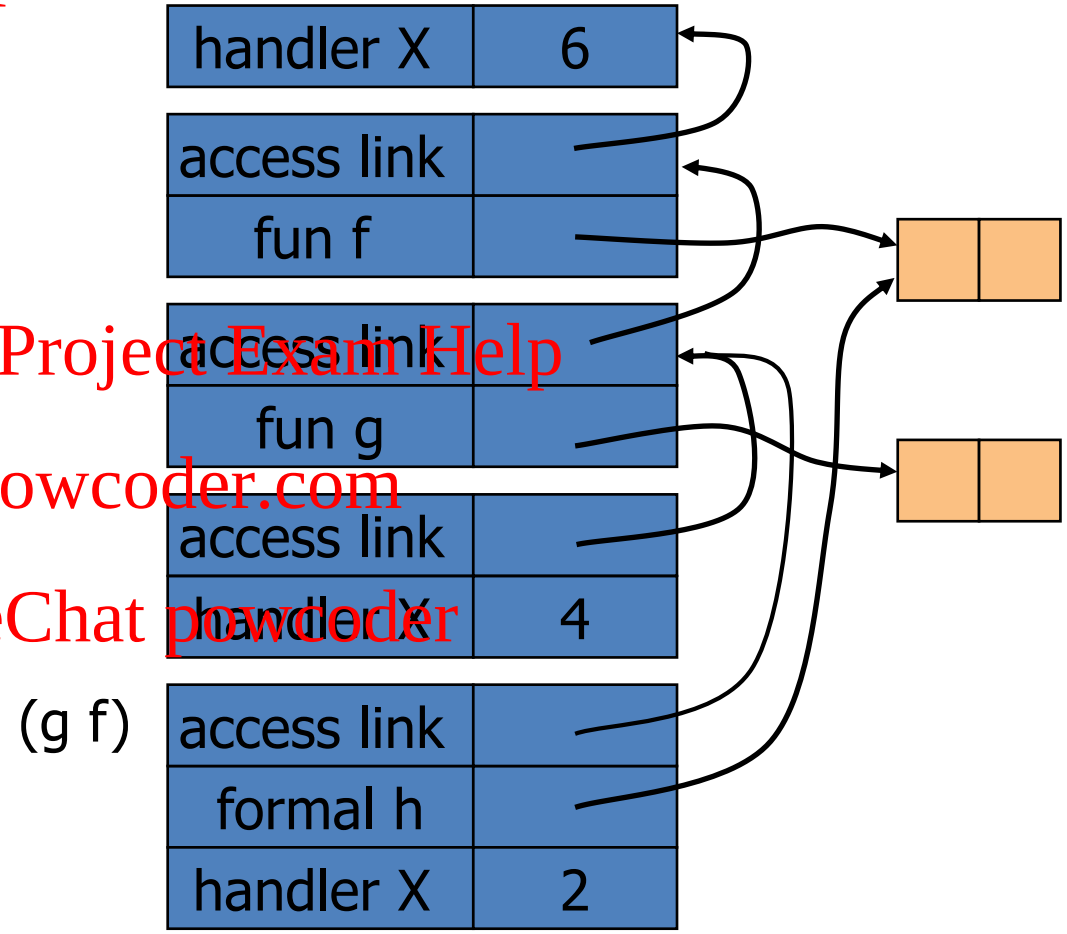
https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

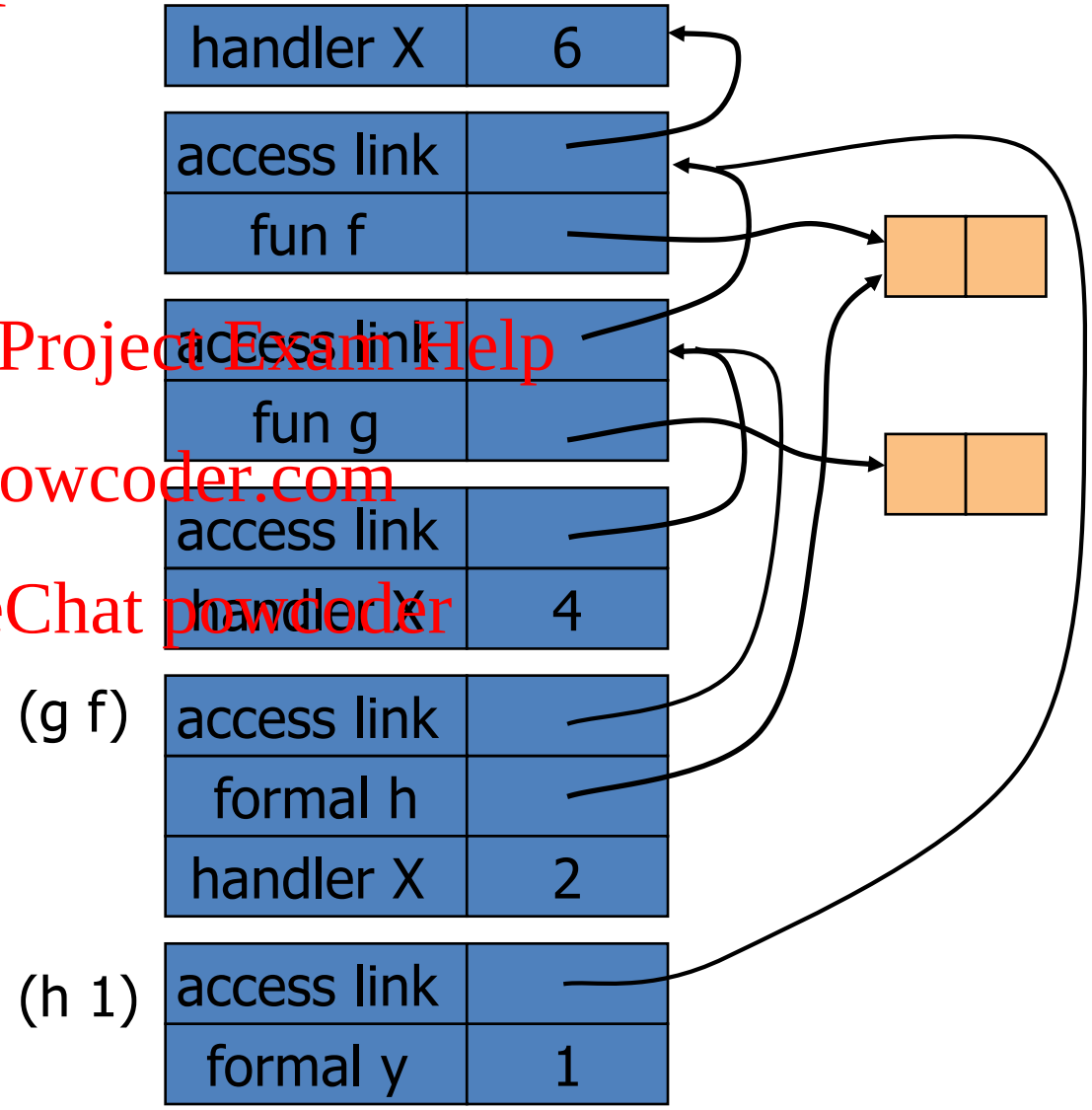
https://powcoder.com

Dynamic Scope of Handler

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

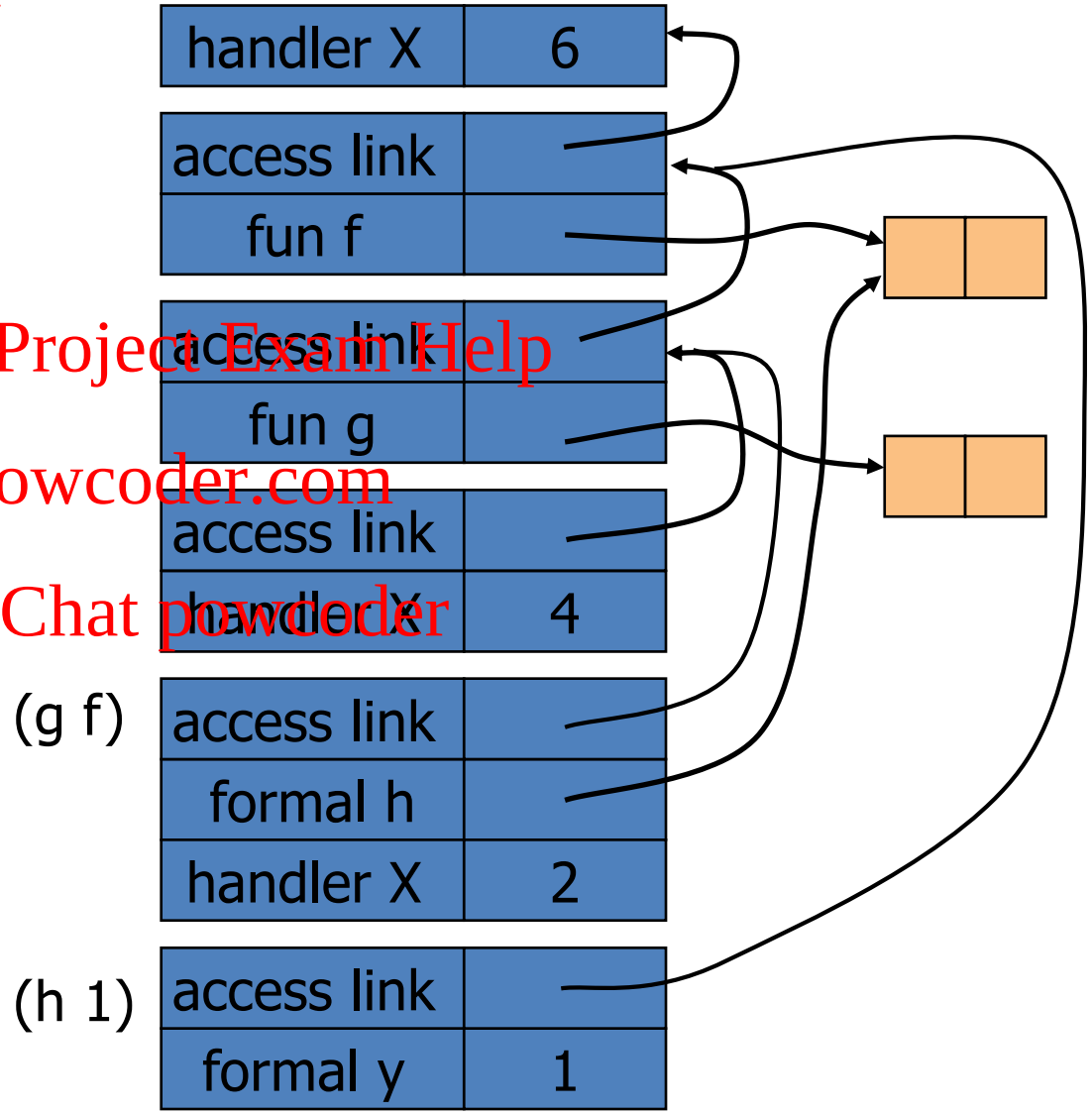
https://powcoder.com
Assignment Project Exam Help

Dynamic Scope of Handler

Add WeChat powcoder

```
exception X
try (let f y = raise X in
     let g h = try (h 1) with X -> 2
     in
     try (g f) with X -> 4)
with X -> 6
```

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



- Dynamic scope: find first X handler, going up the dynamic call chain at the point raise X is executed.
- Result is 2.
- After the handler returns 2, the computation is done, so all activation blocks are popped.

https://powcoder.com

Comparison to Static Scope of Variables

Assignment Project Exam Help

Add WeChat powcoder

```
exception X
try let f y = raise X in
    let g h = try h 1 with X -> 2
    in
    try g f with X -> 4
with X -> 6
```

```
let x = 6 in
let f y = x in
let g h = let x = 2 in
          h 1
in
let x = 4 in
g f
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Static Scope of Declarations

Assignment Project Exam Help

Add WeChat powcoder

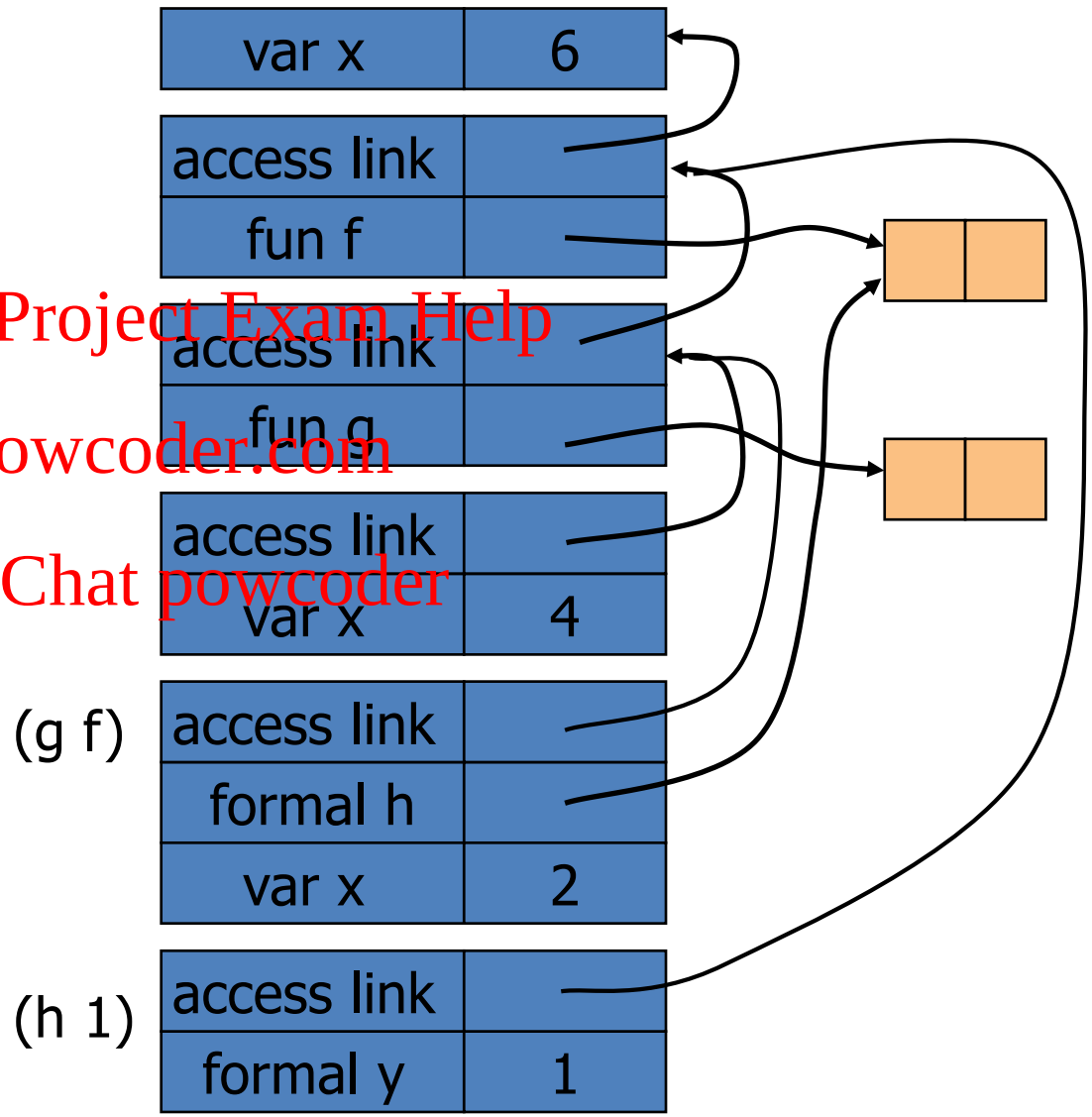
```
let x = 6 in
let f y = x in
let g h = let x = 2 in
          h 1
in
let x = 4 in
g f
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Static scope: find first x, following access links from the reference to x.



https://powcoder.com

Typing of Exceptions

Assignment Project Exam Help

Add WeChat powcoder

- Typing of raise ⟨exn⟩
  - Recall definition of typing
    - Expression e has type t if (normal termination of) e produces value of type t
  - Raising exception is not normal termination
    - Example: 1 + raise X

Assignment Project Exam Help

- Typing of with | ⟨exn⟩ -> ⟨value⟩

https://powcoder.com

  - Converts exception to normal termination

Add WeChat powcoder

  - Need type agreement
  - Examples
    - 1 + (try raise X with X -> e) — Type of e must be int
    - 1 + (try e_1 with X -> e_2) — Type of e_1, e_2 must be int

https://powcoder.com

Exceptions and Resource Allocation

Assignment Project Exam Help
Add WeChat powcoder

```
exception X
try
   (let x = ref [1,2,3]
    in
    let y = ref [4,5,6]
    in
      … raise X
    ) with X -> …
```

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

- Resources may be allocated between handler and raise
- May be “garbage” after exception
- Examples
  - Memory
  - Lock on database
  - Threads
  - …

General problem: no obvious solution

https://powcoder.com

Comparison: ML Example

Assignment Project Exam Help

Add WeChat powcoder

- Exception used to handle a condition that makes it impossible to continue the computation

```
exception Determinant;  (* declare exception name *)
let invert M =                (* function to invert matrix *)
        ...
        if ...
            then raise Determinant    (* exit if Det=0 *)
            else ...
...
in
try invert myMatrix with | Determinant -> ...
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Value for expression if determinant of myMatrix is 0

https://powcoder.com
Assignment Project Exam Help

Comparison: C++ Example

Add WeChat powcoder

```
Matrix invert(Matrix m) {
   if … throw Determinant;
   …
};

try { … invert(myMatrix); …
}
catch (Determinant) { …
   // recover from error
}
```

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

- Note:
  - raise instead of throw
  - catch instead of with
  - try as in ML
- A more significant difference:
  - exceptions are types

https://powcoder.com

Assignment Project Exam Help

Continuations

Add WeChat powcoder

- The main idea:
  - Stop execution, and then later continue

- More precisely:

Assignment Project Exam Help

  - The continuation of an expression in a program is the remaining actions to perform *after* evaluating the expression

https://powcoder.com

Add WeChat powcoder

- Important:
  - does not depend on the expression, only the program that contains it.

https://powcoder.com

Continuations

Assignment Project Exam Help

Add WeChat powcoder

- Idea:
  - The continuation of an expression is "the remaining work to be done after evaluating the expression"
  - Continuation of *e* is a function applied to *e*
- General programming technique
  - Capture the continuation at some point in a program
  - Use it later: "jump" or "exit" by function call
  - A continuation with only a unit argument is like a simple jump.
  - A continuation with arguments is like a jump or exit with data.
- Useful in
  - Compiler optimization: make control flow explicit
  - Operating system scheduling, multiprogramming
  - Web site design

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Example of Continuation Concept

Assignment Project Exam Help

Add WeChat powcoder

- Expression
  - 2*x + 3*y + 1/x + 2/y
- What is continuation of 1/x?
  - Remaining computation after division:

Assignment Project Exam Help

https://powcoder.com

```
let before = 2*x + 3*y in
let continue d = before + d + 2/y
in
continue (1/x)
```

Add WeChat powcoder

  - before is not essential, alternative is:

```
let continue d = 2*x + 3*y + d + 2/y
in
continue (1/x)
```

https://powcoder.com

Example: Error-Avoiding Division using Continuations

Assignment Project Exam Help

Add WeChat powcoder

```
let divide (numer:float) (denom:float)
          (normal_cont: float -> float)
          (error_cont: unit -> float) =
  if denom > 0.0001
  then normal_cont(numer /. denom)
  else error_cont ()
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
let f (x:float) (y:float) =
  let before = 2.0 *. x +. 3.0 *. y in
  let continue (quotient: float) =
      before +. quotient +. 2.0 /. y in
  let error_continue () = before /. 5.2 in
  divide 1.0 x continue error_continue
```

https://powcoder.com

Example: Error Avoiding Division using Exceptions

Assignment Project Exam Help

Add WeChat powcoder

```
exception Div;
let f (x:float) (y:float) =
 try (2.0 *. x +. 3.0 *. y +.
      1.0 /. (if x > 0.0001
              then x
              else raise Div) +.
      2.0 /. y)
 with Div ->
  (2.0 *. x +. 3.0 *. y) /. 5.2
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Same behaviour, simpler with exceptions
- In general, continuations are more flexible than exceptions, but may require more programming effort.

https://powcoder.com

Continuation-Passing Form and Tail Recursion

Assignment Project Exam Help

Add WeChat powcoder

- *continuation-passing form* (CPS): each function or operation is passed a continuation
  - Functions terminate by calling a continuation
  - Thus, no function needs to return to the point from where it was called.
  - Like tail calls…
  - There are systematic rules for transforming an expression or program to CPS.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Assignment Project Exam Help

Example: Tail Recursive Factorial

Add WeChat powcoder

- Standard recursive function

  fact(n) = if n=0 then 1 else n*fact(n-1)

- Tail recursive

  f(n,k) = if n=0 then k else f(n-1, n*k)

  fact(n) = f(n,1)

Assignment Project Exam Help

https://powcoder.com

- How could we derive this?
  - Transform to continuation-passing form
  - Optimize continuation functions to single integer

Add WeChat powcoder

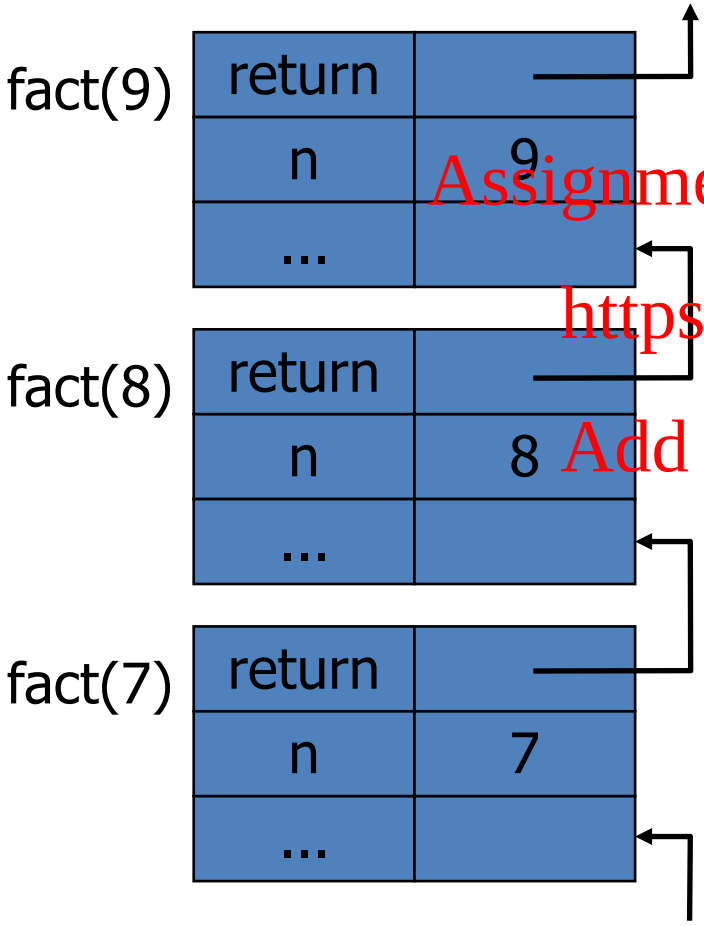
https://powcoder.com

Assignment Project Exam Help

Continuation View of Factorial

Add WeChat powcoder

fact(n) = if n=0 then 1 else n*fact(n-1)



- This invocation multiplies by 9 and returns
- Continuation of fact(8) is λx. 9*x

- Multiplies by 8 and returns
- Continuation of fact(7) is λy. (λx. 9*x) (8*y)

- Multiplies by 7 and returns
- Continuation of fact(6) is λz. (λy. (λx. 9*x) (8*y)) (7*z)

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

https://powcoder.com

Derivation of Tail Recursive Form

Assignment Project Exam Help

Add WeChat powcoder

- Standard function

  fact(n) = if n=0 then 1 else n*fact(n-1)

- Continuation form

  fact(n, k) = if n=0 then k(1)

  Assignment Project Exam Help

  else fact(n-1, λx.k (n*x))

  https://powcoder.com

  fact(n, λx.x) computes n!

  Add WeChat powcoder

- Example computation

  fact(3,λx.x) = fact(2, λy.((λx.x) (3*y)))

  = fact(1, λx.((λy.3*y)(2*x)))

  = fact(0, λy.((λx.3*(2*x))(1*y)))

  = λy.(3*(2*(1*y))) 1 = 6

https://powcoder.com

Derivation of Tail Recursive Form

Assignment Project Exam Help

Add WeChat powcoder

- Continuation-passing form

  fact(n, k) = if n=0 then k(1) else fact(n-1, λx.k (n*x))

- Tail Recursive Form as Optimization of CPS

  fact(n,a) = if n=0 then a else fact(n-1, n*a)

Assignment Project Exam Help

  Each continuation is effectively λx.(a*x) for some a

https://powcoder.com

- Example computation

Add WeChat powcoder

  fact(3,1) = fact(2, 3) was fact(2, λy.3*y)

  = fact(1, 6) was fact(1, λx.6*x)

  = fact(0, 6) = 6

https://powcoder.com

Summary and Other Uses for Continuations

Assignment Project Exam Help

Add WeChat powcoder

- Derivation of Tail Recursive Form (Optimization)
- Explicit Control
  - Normal termination -- call continuation
  - Abnormal termination -- do something else

Assignment Project Exam Help

- Compilation Techniques

https://powcoder.com

  - Call to continuation is functional form of `goto`
  - Continuation-passing style makes control flow explicit

Add WeChat powcoder

- Web Applications and Services (next page)

https://powcoder.com

Web Applications and Services

Assignment Project Exam Help

Add WeChat powcoder

- Web Applications, Web Services, Message-Oriented Middleware (MOM) and Service-Oriented Architecture (SOA) services
  - Handle long running workflows
  - Workflow may take 1 year to complete
  - Progress of subtasks is asynchronous

Assignment Project Exam Help

https://powcoder.com

- Sequential programming is simpler than asynchronous

Add WeChat powcoder

- Continuations provide
  - An easy way to suspend workflow execution at a wait state
  - Thread of control can be resumed when the next message/event occurs, maybe some long time ahead

Continuations supported in some versions of Java JVM

https://powcoder.com

Control of Evaluation Order (Force and Delay)

Assignment Project Exam Help

Add WeChat powcoder

Example: controlling the order for efficiency

let f x y = … x … y … in

f e_1 e_2

- Suppose the value of y is needed only if the value of x has some property.

Assignment Project Exam Help

- Suppose the evaluation of e_2 is expensive.

https://powcoder.com

- We would like:

Add WeChat powcoder

let f x y = … x … Force y … in

f e_1 (Delay e_2)

- where Delay e_2 causes the evaluation of e to be delayed until we call Force (Delay e_2)

https://powcoder.com

Control of Evaluation Order (Force and Delay)

Assignment Project Exam Help

Add WeChat powcoder

- Delay and Force are explicit program constructs in Scheme
- They can be "programmed" in ML.
- Delay e is an abbreviation for (fun () -> e)
  - Example: Delay (3+4) is (fun () -> 3+4)
- Force e is an abbreviation for e()
  - Force (Delay (3+4)) is ((fun () -> 3+4) ()) = 7

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com

Assignment Project Exam Help

Example

Add WeChat powcoder

```
let time_consuming (n:int) =
  let rec tak x y z =
    if x <= y then y
    else tak (tak (x-1) y z) (tak (y-1) z x) (tak (z-1) x y) in
  tak (3*n) (2*n) n;
let rec fib (n:int) =
  if n=0 || n=1 then 1 else fib (n-1) + fib (n-2)
let odd (n:int) = (n mod 2) = 1
let f (x:int) (y:int) = if (odd x) then 1 else (fib y)
in
f (fib 9) (time_consuming 9)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- tak runs for a very long time (and is used by time_consuming)
- Function f has 2 arguments and the second is used only if the first is not odd.

https://powcoder.com

Example (continued)

Assignment Project Exam Help

Add WeChat powcoder

```
let f (x:int) (y:int) = if (odd x) then 1 else (fib y)
in
f (fib 9) (time_consuming 9)
```

- f (fib 9) (time_consuming 9) runs for a very long time
- A version that uses Delay and Force to only evaluate the second argument if needed.

Assignment Project Exam Help

https://powcoder.com

```
let lazy_f (x:int) (y:unit -> int) =
  if odd x then 1 else fib (y())
in
lazy_f (fib 9) (fun () -> time_consuming 9)
```

Add WeChat powcoder

- Because (fib 9) is odd, this expression terminates much more quickly than the one without Delay

https://powcoder.com

Using a Delayed Value More than Once

Assignment Project Exam Help

Add WeChat powcoder

- The version of Delay and Force described so far:
  - Requires static scoping
  - Saves time only if the delayed argument is used at most once.

Assignment Project Exam Help

- A version that works when the delayed argument is used more than once can also be programmed in ML.

https://powcoder.com

- Main idea: store a flag that indicates whether the expression has been evaluated once or not.

Add WeChat powcoder

  - If not, then evaluate when needed and store the result.
  - If so, retrieve the stored result.
  - This is *call-by-need* parameter passing.

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

Implementation and Example

```
type 'a delay =
  | EV of 'a
  | UN of (unit -> 'a)

let ev (d:'a delay) =
  match d with
  | EV x -> x
  | UN f -> f()

let force (d:'a delay ref) =
  let v = ev !d in
  (d := EV v; v)
```

- A delayed value is a reference cell containing an "unevaluated delay"

```
let d = ref (UN (fun () -> fib 9)
```

```
let d =
  ref (UN (fun () ->
             time_consuming 9))
```

- Forcing evaluation evaluates and stores

```
force d = 55
```

- After the call to force:

```
d = ref (EV 55)
```

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

https://powcoder.com

Summary

Assignment Project Exam Help

Add WeChat powcoder

- Exceptions
  - "structured" jumps that may return a value
  - dynamic scoping of exception handler
- Continuations
  - Function representing the rest of the program
  - Generalized form of tail recursion
  - Used in Lisp and ML compilation, some OS projects, web application development, ...
- Delay and Force
  - For controlling evaluation order
  - Can be used to (greatly) improve efficiency
  - Can be used to implement call-by-need parameter passing

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder