

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help
Thinking Inductively

<https://powcoder.com>

Add WeChat powcoder
CSI 3120

Amy Felty

University of Ottawa

<https://powcoder.com> Options

A value v has type t Option if it is either:

- the value `None`, or
- a value `Some v'`, and v' has type t

Options can signal there is no useful result to the computation

<https://powcoder.com>

Example: we look up a value in a hash table using a key.

- If the key is present, return `Some v` where v is the associated value
- If the key is not present, we return `None`

Slope between two points

Assignment Project Exam Help

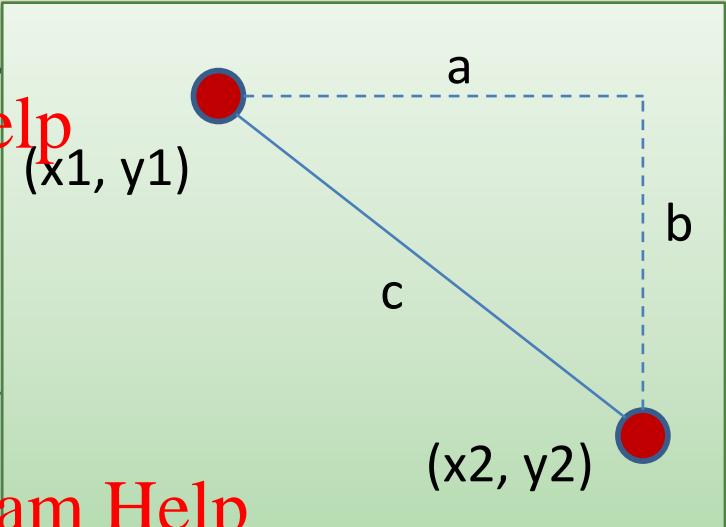
Add WeChat powcoder

```
type point = float * float
```

Assignment Project Exam Help

```
let slope (p1:point) (p2:point) : float =
```

Add WeChat powcoder



Slope between two points

Assignment Project Exam Help

Add WeChat powcoder

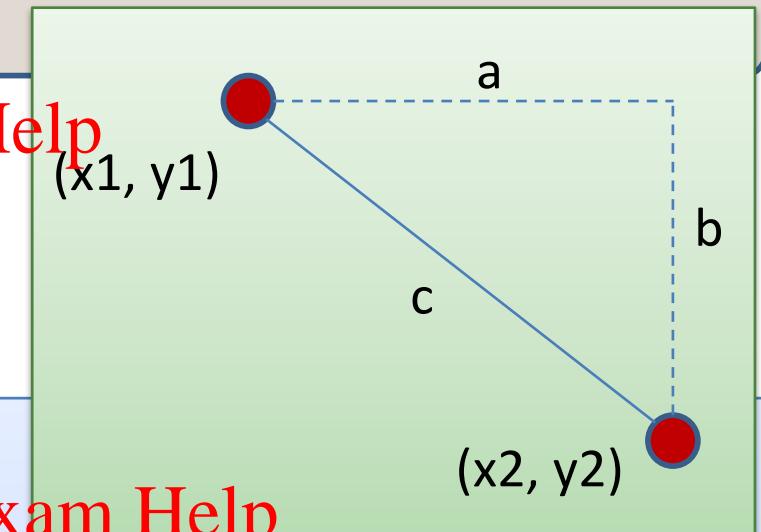
```
type point = float * float
```

Assignment Project Exam Help

```
let slope (p1:point) (p2:point) : float =  
  let (x1,y1) = p1 in  
  let (x2,y2) = p2 in
```

Add WeChat powcoder

deconstruct tuple



Slope between two points

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

Add WeChat powcoder

```
type point = float * float
```

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

```
let slope (p1:point) (p2:point) : float =
    let (x1, y1) = p1 in
```

```
    let (x2, y2) = p2 in
```

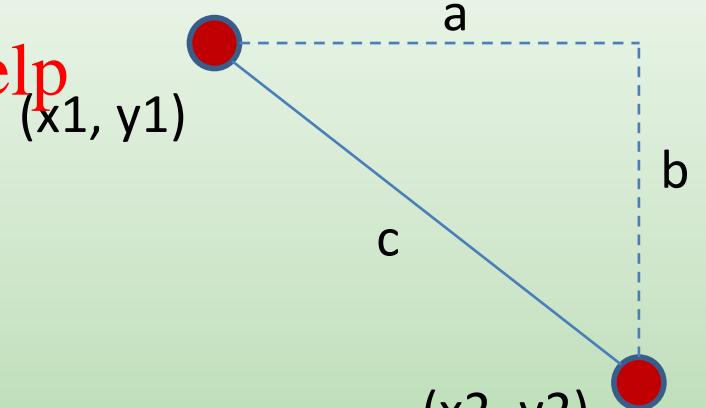
```
    let xd = x2 -. x1 in
```

```
    if xd != 0.0 then
```

```
        (y2 -. y1) /. xd
```

```
    else
```

```
        ???
```



← avoid divide by zero

what can we return?

Slope between two points

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

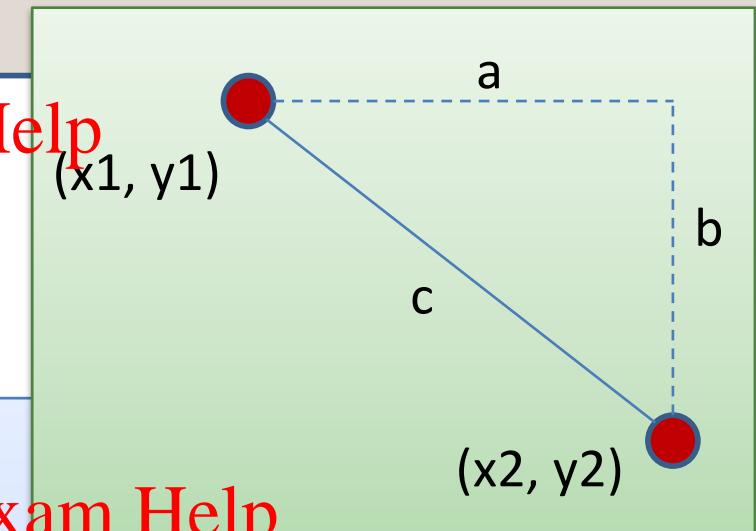
[Add WeChat powcoder](#)

```
type point = float * float
```

[Assignment Project Exam Help](#)

```
let slope (p1:point) (p2:point) : float option =
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  let xd = x2 -. x1 in
  if xd != 0.0 then
    ???
  else
    ???
```

[we need an option type as the result type](#)

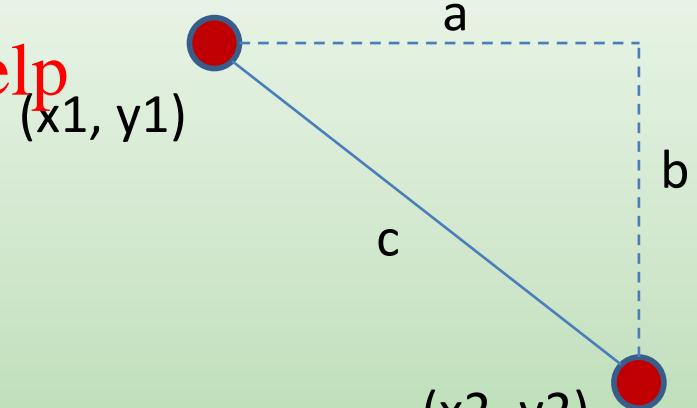


Slope between two points

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float
```



```
let slope (p1:point) (p2:point) : float option =
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  let xd = x2 -. x1 in
  if xd != 0.0 then
    Some ((y2 -. y1) /. xd)
  else
    None
```

Slope between two points

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

Add WeChat powcoder

```
type point = float * float
```

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

```
let slope (p1:point) (p2:point) : float option =
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  let xd = x2 -. x1 in
  if xd != 0.0 then
```

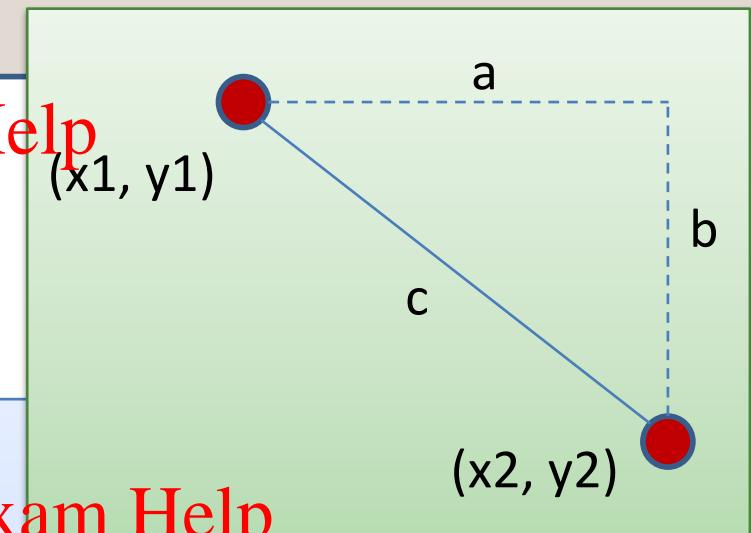
$\frac{(y2 - y1)}{xd}$

else

None

Has type float

Can have type float option



Slope between two points

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

Add WeChat powcoder

```
type point = float * float
```

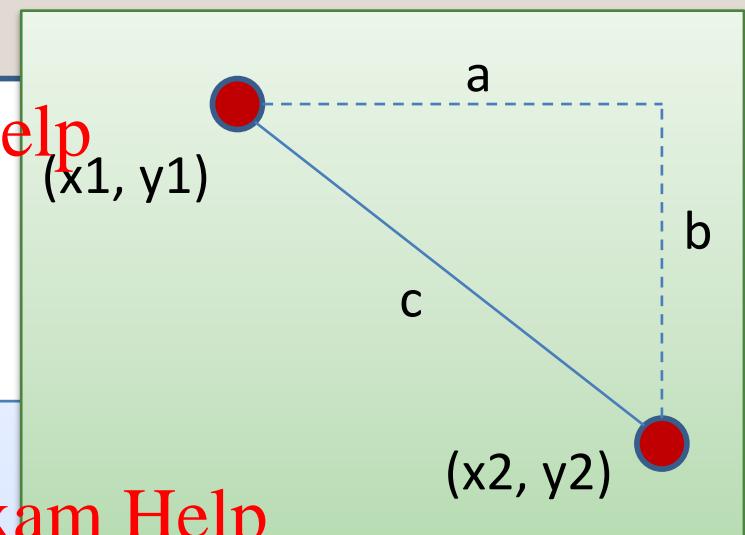
[Assignment Project Exam Help](https://powcoder.com)

```
let slope (p1:point) (p2:point) : float option =
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  let xd = x2 -. x1 in
  if xd != 0.0 then
    (y2 -. y1) /. xd
  else
    None
```

Has type **float**

WRONG: Type mismatch

Can have type **float option**



Slope between two points

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

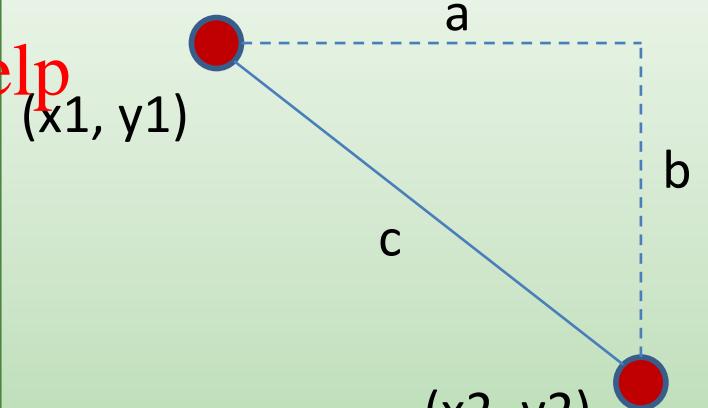
Add WeChat powcoder

```
type point = float * float
```

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

```
let slope (p1:point) (p2:point) : float option =
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  let xd = x2 -. x1 in
  if xd != 0.0 then
    (y2 -. y1) /. xd
  else
    None
```

Has type **float**



doubly WRONG:
result does not
match declared result

<https://powcoder.com> Remember the typing rule for if

Assignment Project Exam Help

Add WeChat powcoder

if e1 : bool

and e2 : t and e3 : t (for some type t)

then if e1 then e2 else e3 : t

Assignment Project Exam Help

Returning <https://powcoder.com> an optional value from an if statement:

Add WeChat powcoder

if ... then

None : t option

else

Some (...) : t option

<https://powcoder.com> How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

returns a float option

<https://powcoder.com> How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

let print_slope (p1:point) (p2:point) : unit =

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com> How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

let print_slope (p1:point) (p2:point) : unit =
 slope p1 p2

<https://powcoder.com>

Add WeChat powcoder

returns a float option;
to print we must discover if it is
None or Some

<https://powcoder.com> How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

let print_slope (p1:point) (p2:point) : unit =
match slope p1 p2 with
<https://powcoder.com>

Add WeChat powcoder

How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

let print_slope (p1:point) (p2:point) : unit =
 match slope p1 p2 with

Some s ->

| None ->

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

There are two possibilities

Vertical bar separates possibilities

How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

```
let print_slope (p1:point) (p2:point) : unit =
  match slope p1 p2 with
    Some s =>
      Add WeChat powcoder
    | None ->
      Add WeChat powcoder
```

The "Some s" pattern includes the variable s

The object between | and -> is called a pattern

https://powcoder.com How do we use an option?

Assignment Project Exam Help

Add WeChat powcoder

slope : point -> point -> float option

Assignment Project Exam Help
let print_slope (p1:point) (p2:point) : unit =
 match slope p1 p2 with
 Some s ->
 print_string ("Slope: " ^ string_of_float s)
 | None ->
 print_string "Vertical line.\n"

Writing Functions Over Typed Data

- Steps to writing functions over typed data:

1. Write down the function and argument names
2. Write down argument and result types
3. Write down some examples (in a comment)
4. Deconstruct input data structures
5. Build new output values
6. Clean up by identifying repeated patterns

- For option types:

when the **input** has type **t option**,
deconstruct with:

```
match ... with
| None -> ...
| Some s -> ...
```

when the **output** has type **t option**,
construct with:

Some (...)

None

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

MORE PATTERN MATCHING

Recall the Distance Function

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
    let square x = x *. x in
    let (x1, y1) = p1 in
    let (x2, y2) = p2 in
    sqrt (square (x2 -. x1) + square (y2 -. y1))
```

Recall the Distance Function

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
    let square x = x *. x in
    let (x1, y1) = p1 in
    let (x2, y2) = p2 in
    sqrt (square (x2 -. x1) + square (y2 -. y1))
;;
```

(x_2, y_2) is an example of a pattern – a pattern for tuples.

So let declarations can contain patterns just like match statements

The difference is that a match allows you to consider multiple different data shapes

Recall the Distance Function

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
  let square x = x *. x in
  match p1 with
  | (x1, y1) -> https://powcoder.com
    let (x2, y2) = p2 in
    sqrt (square (x2 -. x1) +. square (y2 -. y1))
;;
```



There is only 1 possibility when matching a pair

Recall the Distance Function

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
    let square x = x *. x in
    match p1 with
    | (x1, y1) -> https://powcoder.com
        match p2 with
        | (x2, y2) ->
            sqrt (square (x2 -. x1) +. square (y2 -. y1))
;;
```



We can nest one match expression inside another.

(We can nest any expression inside any other, if the expressions have the right types)

<https://powcoder.com>
Complex Patterns

Assignment Project Exam Help
we built a pair of pairs

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
    let square x = x *. x in
    match (p1, p2) with
    | ((x1, y1), (x2, y2)) ->
        sqrt (square (x2 -. x1) +. square (y2 -. y1))
;;
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Pattern for a pair of pairs: **((variable, variable), (variable, variable))**
All the variable names in the pattern must be different.

Complex Patterns

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance (p1:point) (p2:point) : float =
    let square x = x *. x in
    match (p1, p2) with
    | ((x1, y1), (x2, y2)) ->
        sqrt (square (x2 -. x1) +. square (y2 -. y1))
;;
```



A pattern must be **consistent with** the type of the expression
in between **match ... with**

For example, $x_1 : \text{float}$, $x_2 : \text{float}$, $p1 : \text{float} * \text{float}$

Pattern-matching in function parameters

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

let distance ((x1,y1):point) ((x2,y2):point) : float =
  let square x = x *. x in
    sqrt (square (x2 -. x1) + square (y2 -. y1))
;;
```

Add WeChat powcoder

Function parameters are patterns too!

<https://powcoder.com> What's the best style?

Assignment Project Exam Help

```
let distance (p1:point) (p2:point) : float =  
    let square x = x *. x in  
    let (x1,y1) = p1 in  
    let (x2,y2) = p2 in  
    sqrt (square (x2 -. x1) +. square (y2 -. y1))
```

<https://powcoder.com>

```
let distance ((x1,y1):point) ((x2,y2):point) : float =  
    let square x = x *. x in  
    sqrt (square (x2 -. x1) +. square (y2 -. y1))
```

Either of these is reasonably clear and compact.

Code with unnecessary nested matches/lets is particularly ugly to read.

<https://powcoder.com> What's the best style?

Assignment Project Exam Help

Add WeChat powcoder

```
let distance (x1, y1) (x2, y2) =  
    let square x = x * x in  
    sqrt (square (x2 -. x1) +. square (y2 -. y1))
```

<https://powcoder.com>

Add WeChat powcoder

Note that the types for tuples plus the tuple patterns are a little ugly/verbose, and they can be inferred from the operators used such as `+'. ... but for now in class, use the explicit type annotations.

https://powcoder.com Other Patterns

Assignment Project Exam Help

Add WeChat powcoder

```
type point = float * float

(* returns a nearby point in the graph if one exists *)
nearby : graph -> point -> point option

let printer (g:graph) (p:point) : unit =
  match nearby g p with
  | None -> print_string "could not find one\n"
  | Some (x,y) ->
    print_float x;
    print_string ", ";
    print_float y;
    print_newline();
;;

```

https://powcoder.com Other Patterns

- Constant values can be used as patterns

Add WeChat powcoder

```
let small_prime (n:int) : bool =  
  match n with  
  | 2 -> true  
  | 3 -> true  
  | 5 -> true  
  | _ -> false  
;;
```

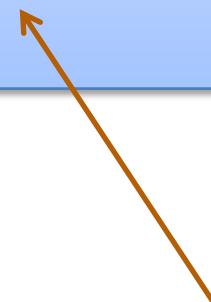
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
let iffy (b:bool) : int =  
  match b with  
  | true -> 0  
  | false -> 1  
;;
```

the underscore pattern
matches anything
it is the "don't care" pattern



<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

INDUCTIVE THINKING

Inductive Programming and Proving

An *inductive data type* is a data type defined by:

- a collection of base cases
 - that don't refer to T
- a collection of inductive cases that build new values of type T from pre-existing data of type T
 - the pre-existing data is guaranteed to be *smaller* than the new values

Programming principle: <https://powcoder.com>

- solve programming problem for base cases
- solve programming problem for inductive cases by calling function recursively (inductively) on *smaller* data value

Proving principle:

- prove program satisfies property P for base cases
- prove inductive cases satisfy property P assuming inductive calls on *smaller* data values satisfy property P

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

LISTS: AN INDUCTIVE DATA TYPE

Lists are Recursive Data

- In OCaml, a list value is:

- `[]` (the empty list)
- `v :: vs` (a value `v` followed by a shorter list of values `vs`)



Lists are Inductive Data

- In OCaml, a list value is:
 - [] (the empty list)
 - $v :: vs$ (a value v followed by a shorter list of values vs)
- An example:
 - $2 :: 3 :: 5 :: []$ has type int list
 - is the same as: $2 :: (3 :: (5 :: []))$
 - ":" is called "cons"
- An alternative syntax ("syntactic sugar" for lists):
 - [2; 3; 5]
 - But this is just a shorthand for $2 :: 3 :: 5 :: []$. If you ever get confused fall back on the 2 basic *constructors*: :: and []

<https://powcoder.com>
Typing Lists

- Typing Assignment Project Exam Help
Rules for lists:

- (1) Add WeChat powcoder
[] may have any list type t list
- (2) if e1 : t and e2 : t list
then (e1 :: e2) : t list

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com>
Typing Lists

- Typing Assignment Project Exam Help
Rules for lists:

(1) Add WeChat powcoder
[] may have any list type t list

(2) if e1 : t and e2 : t list
then (e1 :: e2) : t list

<https://powcoder.com>

- More examples:

(1 + 2) :: (3 + 4) :: [] : ??

(2 :: []) :: (5 :: 6 :: []) :: [] : ??

[[2]; [5; 6]] : ??

https://powcoder.com Typing Lists

- Typing Assignment Project Exam Help

(1) Add WeChat powcoder
[] may have any list type t list

(2) if e1 : t and e2 : t list
then (e1 :: e2) : t list

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

- More examples:

(1 + 2) :: (3 + 4) :: [] : int list

(2 :: []) :: (5 :: 6 :: []) :: [] : (int list) list or just int list list

[[2]; [5; 6]] : int list list

(Remember that the 3rd example is an abbreviation for the 2nd)

<https://powcoder.com>
Another Example

Assignment Project Exam Help

- What type does this have?

Add WeChat powcoder

[2] :: [3]

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Another Example

Assignment Project Exam Help

- What type does this have?

Add WeChat powcoder

[2] :: [3]
Assignment Project Exam Help
int list https://powcoder.com int list

Add WeChat powcoder

```
# [2] :: [3];;
```

Error: This expression has type int but an
expression was expected of type
int list

```
#
```

<https://powcoder.com>
Another Example

Assignment Project Exam Help

- What type does this have?

Add WeChat powcoder

[2] :: [3]
Assignment Project Exam Help
int list <https://powcoder.com> int list

Add WeChat powcoder

- What is a simple fix that makes the expression type check?

Another Example

Assignment Project Exam Help

- What type does this have?

Add WeChat powcoder

[2] :: [3]
Assignment Project Exam Help
int list https://powcoder.com int list

Add WeChat powcoder

- What is a simple fix that makes the expression type check?

Either: 2 :: [3] : int list

Or: [2] :: [[3]] : int list list

https://powcoder.com Analyzing Lists

- Just like options, there are two possibilities when deconstructing lists. Hence we use a match with two branches

Add WeChat powcoder

```
(* return Some v, if v is the first list element;
   return None, if the list is empty *)  
https://powcoder.com  
let head (xs : int list) : int option =  
Add WeChat powcoder
```

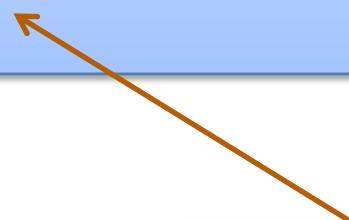
https://powcoder.com Analyzing Lists

- Just like **Assignment Project Exam Help**, there are two possibilities when deconstructing lists. Hence we use a match with two branches
Add WeChat powcoder

```
(* return Some v, if v is the first list element;  
   return None, if the list is empty *)
```

https://powcoder.com

```
let head (xs : int list) : int option =  
  match xs with  
  | [] ->  
  | hd :: _ ->
```



we don't care about the contents of the tail of the list so we use the underscore

https://powcoder.com Analyzing Lists

- Just like options, there are two possibilities when deconstructing lists. Hence we use a match with two branches

Add WeChat powcoder

```
(* return Some v, if v is the first list element;
   return None, if the list is empty *)
```

https://powcoder.com

```
let head (xs : int list) : int option =
  match xs with
  | [] -> None
  | hd :: _ -> Some hd
```

- This function isn't recursive -- we only extracted a small , fixed amount of information from the list -- the first element

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [ (2,3), (4,7), (5,2) ] == [6, 28; 10]  
*)
```

<https://powcoder.com>

Add WeChat powcoder

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

prods [(2,3), (4,7), (5,2)] = [6, 28; 10]

*)

<https://powcoder.com>

let rec prods (xs : (int * int) list) : int list =

Add WeChat powcoder

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

<https://powcoder.com>

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] ->  
    | (x,y) :: tl ->
```

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

Assignment Project Exam Help

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] -> []  
    | (x,y) :: tl ->
```

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

https://powcoder.com

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] -> []  
    | (x,y) :: tl -> ?? :: ??
```

the result type is int list, so we can speculate
that we should create a list

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

<https://powcoder.com>

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] -> []  
    | (x,y) :: tl -> (x * y) :: ??
```

the first element is the product

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

<https://powcoder.com>

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] -> []  
    | (x,y) :: tl -> (x * y) :: ??
```



to complete the job, we must compute
the products for the rest of the list

A more interesting example

Assignment Project Exam Help

Add WeChat powcoder

(* Given a list of pairs of integers,
produce the list of products of the pairs

```
prods [(2,3); (4,7); (5,2)] = [6; 28; 10]  
*)
```

<https://powcoder.com>

```
let rec prods (xs : (int * int) list) : int list =  
  match xs with  
    | [] -> []  
    | (x,y) :: tl -> (x * y) :: prods tl
```

Three Parts to Constructing a Function

Assignment Project Exam Help

(1) Think about how to *break down* the input into cases:

```
let rec prods (xs : int * int list) : int list =
  match xs with
  | [] -> []
  | (x, y) :: tl -> ... prods tl ...
```

Add WeChat powcoder

This assumption is called the Induction Hypothesis. It can be used it to prove your program correct.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

(2) *Assume* the recursive call on smaller data is correct.

(3) Use the result of the recursive call to *build* correct answer.

```
let rec prods (xs : (int * int) list) : int list =
  ...
  | (x, y) :: tl -> ... prods tl ...
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

(* Given two lists of integers,
return None if the lists are different lengths
otherwise stitch the lists together to create
Some of a list of pairs

<https://powcoder.com>

```
zip [2; 3] [4; 5] == Some [(2,4); (3,5)]  
zip [5; 3] [4] == None  
zip [4; 5; 6] [8; 9; 10; 11; 12] == None  
*)
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
    : (int * int) list option =
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
    : (int * int) list option =
    match (xs, ys) with
```

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
```

```
match (xs, ys) with
| ([], []) -> https://powcoder.com
| ([], y :: ys') ->
| (x :: xs', []) -> Add WeChat powcoder
| (x :: xs', y :: ys') ->
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
```

```
match (xs, ys) with
| ([], []) -> Some []
| ([], y :: ys') ->
| (x :: xs', []) -> Add WeChat powcoder
| (x :: xs', y :: ys') ->
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
```

```
match (xs, ys) with
| ([], []) -> Some []
| ([], y :: ys') -> None
| (x :: xs', []) -> None
| (x :: xs', y :: ys') ->
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
  match (xs, ys) with
  | ([], []) -> Some []
  | ([], y :: ys') -> None
  | (x :: xs', []) -> None
  | (x :: xs', y :: ys') -> (x, y) :: zip xs' ys'
```



is this ok?

https://powcoder.com
Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
  match (xs, ys) with
  | ([], []) -> Some []
  | ([], y :: ys') -> None
  | (x :: xs', []) -> None
  | (x :: xs', y :: ys') -> (x, y) :: zip xs' ys'
```

No! zip returns a list option, not a list!
We need to match it and decide if it is Some or None.



https://powcoder.com
Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
  match (xs, ys) with
  | ([], []) -> Some []
  | ([], y :: ys') -> None
  | (x :: xs', []) -> None
  | (x :: xs', y :: ys') ->
    (match zip xs' ys' with
     None -> None
     | Some zs -> (x, y) :: zs)
```



Is this ok?

Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
  match (xs, ys) with
  | ([], []) -> Some []
  | ([], y :: ys') -> None
  | (x :: xs', []) -> None
  | (x :: xs', y :: ys') ->
    (match zip xs' ys' with
     None -> None
     | Some zs -> Some ((x, y) :: zs))
```

<https://powcoder.com> Another example: zip

Assignment Project Exam Help

Add WeChat powcoder

```
let rec zip (xs : int list) (ys : int list)
  : (int * int) list option =
  match (xs, ys) with
  | ([], []) -> Some []
  | (x :: xs', y :: ys') ->
    (match zip xs' ys' with
     | None -> None
     | Some zs -> Some ((x, y) :: zs))
  | (_, _) -> None
```

Assignment Project Exam Help

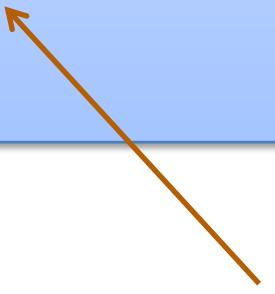
Add WeChat powcoder

Add WeChat powcoder

None -> None

| Some zs -> Some ((x, y) :: zs))

| (_, _) -> None



Clean up.

Reorganize the cases.

Pattern matching proceeds in order.

<https://powcoder.com>
A bad list example

Assignment Project Exam Help

Add WeChat powcoder

```
let rec sum (xs : int list) : int =  
  match xs with  
  | hd :: tl -> hd + sum tl
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com>
A bad list example

Assignment Project Exam Help

Add WeChat powcoder

```
let rec sum (xs : int list) : int =
  match xs with
  | hd :: tl -> hd + sum tl
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
#          Characters 39-78:
..match xs with
    hd :: tl -> hd + sum tl..
```

Warning 8: this pattern-matching is not exhaustive.

Here is an example of a value that is not matched: []

```
val sum : int list -> int = <fun>
```

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

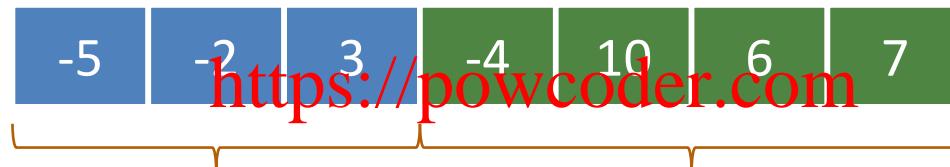
INSERTION SORT

Recall Insertion Sort

Assignment Project Exam Help

- At any point during the insertion sort:
 - some initial segment of the array will be sorted
 - the rest of the array will be in the same (unsorted) order as it was originally

Assignment Project Exam Help



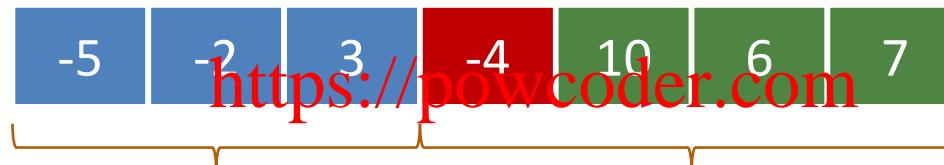
sorted unsorted

Recall Insertion Sort

Assignment Project Exam Help

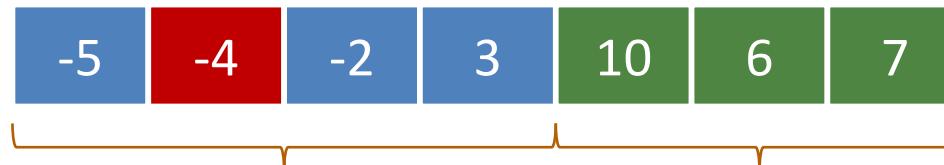
- At any point during the insertion sort:
 - some initial segment of the array will be sorted
 - the rest of the array will be in the same (unsorted) order as it was originally

Assignment Project Exam Help



Add WeChat powcoder

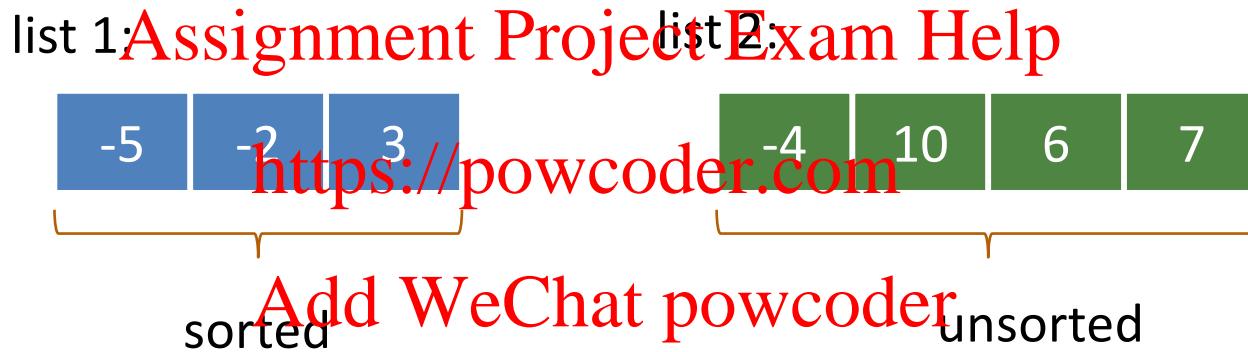
- At each step, take the next item in the array and insert it in order into the sorted portion of the list



sorted unsorted

https://powcoder.com Insert Sort With Lists

- The algorithm is similar, except instead of *one array*, we will maintain *two lists*, a sorted list and an unsorted list



- We'll factor the algorithm:
 - a function to insert into a sorted list
 - a sorting function that repeatedly inserts

https://powcoder.com
Insert

Assignment Project Exam Help

Add WeChat powcoder

(* insert x in to sorted list xs *)

let rec insert (x : int) (xs : int list) : int list =

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://powcoder.com
Insert

Assignment Project Exam Help

Add WeChat powcoder

```
(* insert x in to sorted list xs *)
```

```
let rec insert (x : int) (xs : int list) : int list =
  match xs with
  | [] ->
  | hd :: tl -> https://powcoder.com
```

Add WeChat powcoder

a familiar pattern:
analyze the list by cases

https://powcoder.com
Insert

Assignment Project Exam Help

Add WeChat powcoder

(* insert x in to sorted list xs *)

```
let rec insert (x : int) (xs : int list) : int list =  
  match xs with  
    | [] -> [x] ←   
    | hd :: tl -> https://powcoder.com  
      insert x into the  
      empty list
```

Add WeChat powcoder

https://powcoder.com
Insert

Assignment Project Exam Help

Add WeChat powcoder

```
(* insert x in to sorted list xs *)
```

```
let rec insert (x : int) (xs : int list) : int list =  
  match xs with
```

```
  | [] -> [x]  
  | hd :: tl -> if hd < x then  
    hd :: insert x tl
```

Assignment Project Exam Help

build a new list with:

- hd at the beginning
- the result of inserting x in to the tail of the list afterwards

https://powcoder.com
Insert

Assignment Project Exam Help

Add WeChat powcoder

```
(* insert x in to sorted list xs *)
```

```
let rec insert (x : int) (xs : int list) : int list =
  match xs with
    | [] -> [x]
    | hd :: tl ->
      if hd < x then
        hd :: insert x tl
      else
        x :: xs
```

put x on the front of the list,
the rest of the list follows

https://powcoder.com Insertion Sort

Assignment Project Exam Help

```
type il = int list
```

```
insert : int -> il -> il
```

```
(* insertion sort *)
```

```
let rec insert_sort =
```

Add WeChat powcoder

https://powcoder.com Insertion Sort

Assignment Project Exam Help

```
type il = int list
```

```
insert : int -> il -> il
```

```
(* insertion sort *)
```

```
let rec insert_sort (xs : il) : il
```

```
let rec aux (sorted : il) (x : int) : il =
```

```
in
```

https://powcoder.com Insertion Sort

Assignment Project Exam Help

```
type il = int list
```

```
insert : int -> il -> il
```

```
(* insertion sort *)
```

```
let rec insert_sort (xs : il) : il
```

```
let rec aux (sorted : il) (unsorted : il) : il =
```

```
in
```

```
aux [] xs
```

https://powcoder.com Insertion Sort

Assignment Project Exam Help

```
type il = int list
```

```
insert : int -> il -> il
```

```
(* insertion sort *)
```

```
let rec insert_sort (xs : il) : il
```

```
let rec aux (sorted : il) (unsorted : il) : il =
  match unsorted with
  | [] ->
  | hd :: tl ->
    in
    aux [] xs
```

https://powcoder.com Insertion Sort

Assignment Project Exam Help

```
type il = int list
```

```
insert : int -> il -> il
```

```
(* insertion sort *)
```

```
let rec insert_sort (xs : il) : il
```

```
let rec aux (sorted : il) (unsorted : il) : il =
  match unsorted with
  | [] -> sorted
  | hd :: tl -> aux (insert hd sorted) tl
in
aux [] xs
```

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

A SHORT JAVA RANT

https://powcoder.com Definition and Use of Java Pairs

Assignment Project Exam Help

```
public class Pair {  
  
    public int x;  
    public int y;  
  
    public Pair (int a,int b) {  
        x = a;  
        y = b;  
    }  
}
```

```
public class User {  
  
    public Pair swap (Pair p1) {  
        Pair p2 =  
            new Pair(p1.y, p1.x);  
        return p2;  
    }  
}
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

What could go wrong?

A Paucity of Types

Assignment Project Exam Help

```
public class Pair {  
  
    public int x;  
    public int y;  
  
    public Pair (int a, int b) {  
        x = a;  
        y = b;  
    }  
}
```

Add WeChat powcoder

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

```
public class User {  
  
    public Pair swap (Pair p1) {  
        Pair p2 =  
            new Pair(p1.y, p1.x);  
        return p2;  
    }  
}
```

Add WeChat powcoder

The input **p1** to swap may be **null** and we forgot to check.

Java has no way to define a pair data structure that is *just a pair*.

How many students in the class have seen an accidental null pointer exception thrown in their Java code?

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

In OCaml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

In OCaml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

Assignment Project Exam Help

And if you write code like this:

<https://powcoder.com>

```
let swap_java_pair (p:java_pair) : java_pair =
  let (x,y) = p in
```

Add WeChat powcoder

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

In OCaml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

Assignment Project Exam Help

And if you write code like this:

<https://powcoder.com>

```
let swap_java_pair (p:java_pair) : java_pair =
  let (x,y) = p in
    (y,x)
```

You get a *helpful* error message like this:

```
# ... Characters 91-92:
```

```
let (x,y) = p in (y,x);;
```

```
Error: This expression has type java_pair = (int * int) option
but an expression was expected of type 'a * 'b
```

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

```
type java_pair = (int * int) option
```

And what if you were up late trying to finish your assignment and you accidentally wrote the following statement?

<https://powcoder.com>

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
    | Some (x,y) -> Some (y,x)
```

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

```
type java_pair = (int * int) option
```

And what if you were up late trying to finish your assignment and you accidentally wrote the following statement?

<https://powcoder.com>

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
    | Some (x,y) -> Add WeChat powcoder
```

OCaml to the rescue!

```
..match p with
  | Some (x,y) -> Some (y,x)
```

Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
None

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

```
type java_pair = (int * int) option
```

And what if you were up late trying to finish your assignment and you accidentally wrote the following statement?

<https://powcoder.com>

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
    | Some (x,y) -> Some (y,x)
```



```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
    | None -> None
    | Some (x,y) -> Some (y,x)
```

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

Add WeChat powcoder

Moreover, your pairs are probably almost never null!

Assignment Project Exam Help

<https://powcoder.com>

Defensive programming & always checking for null can be annoying.

From Java Pairs to OCaml Pairs

Assignment Project Exam Help

Add WeChat powcoder

There just isn't always some "good thing" for a function to do when it receives a bad input, like a null pointer

Assignment Project Exam Help

In OCaml, all these issues disappear when you use the proper type for a pair and that type contains no "extra junk"

```
type pair = int * int
```

Add WeChat powcoder

Once you know OCaml, it is *hard* to write swap incorrectly

Your *bullet-proof* code is much simpler than in Java.

```
let swap (p:pair) : pair =
  let (x,y) = p in (y,x)
```

https://powcoder.com Summary of Java Pair Rant

Assignment Project Exam Help

Java has a paucity of types

- There is no type to describe just the pairs
- There is no type to describe just the triples
- There is no type to describe the pairs of pairs
- There is no type ...

OCaml has many more types

- use option when things may be null
- do not use option when things are not null
- OCaml types describe data structures more precisely
 - programmers have fewer cases to worry about
 - entire classes of errors just go away
 - type checking and pattern analysis help prevent programmers from ever forgetting about a case

https://powcoder.com Summary of Java Pair Rant

Assignment Project Exam Help

Java has a paucity of type annotations.

- There is no type to describe just the pairs
- There is no type to describe just the tuples
- There is no type to describe just the lists
- There is no type to describe just the sets

OCaml has many type annotations.

- uses type annotations
- does type annotations

https://powcoder.com
SCORE: OCAML 1, JAVA 0
Add WeChat powcoder

• type annotations help prevent programmers from ever forgetting about a case

<https://powcoder.com>
C, C++ Rant

Assignment Project Exam Help

Java has a paucity of types
[Add WeChat powcoder](#)

- but at least when you forget something,
it ***throws an exception*** instead of ***silently going off the rails!***

[Assignment Project Exam Help](#)

If you forget to check <https://powcoder.in> program,

- no type-check error at compile time
- no exception at run time
- it might crash right away (that would be best), or
- it might permit a buffer-overrun (or similar) vulnerability

Summary of C, C++ rant

Assignment Project Exam Help

Java has a paucity of types

- but at least when you forget something it **throws an exception** instead of crashing off a trolley!

Assignment Project Exam Help

If you

- no type

https://powcoder.com

SCORE:

OCAML 1, JAVA 0, C -1

- it's not checked

- it's not checked

- so the hacker can do anything

– it's not checked

– similar vulnerability