Assignment Project Exam Help

COMP90038
Algorithms and Complexity

https://powcoder.com

Lecture 2: Review of Basic Concepts

(with thanks to Harald Søndergaard)

Add WeChat powcoder

Toby Murray

toby.murray@unimelb.edu.au

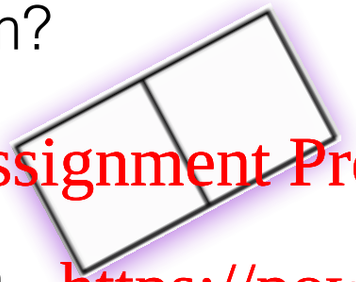
DMD 8.17 (Level 8, Doug McDonell Bldg)

http://people.eng.unimelb.edu.au/tobym

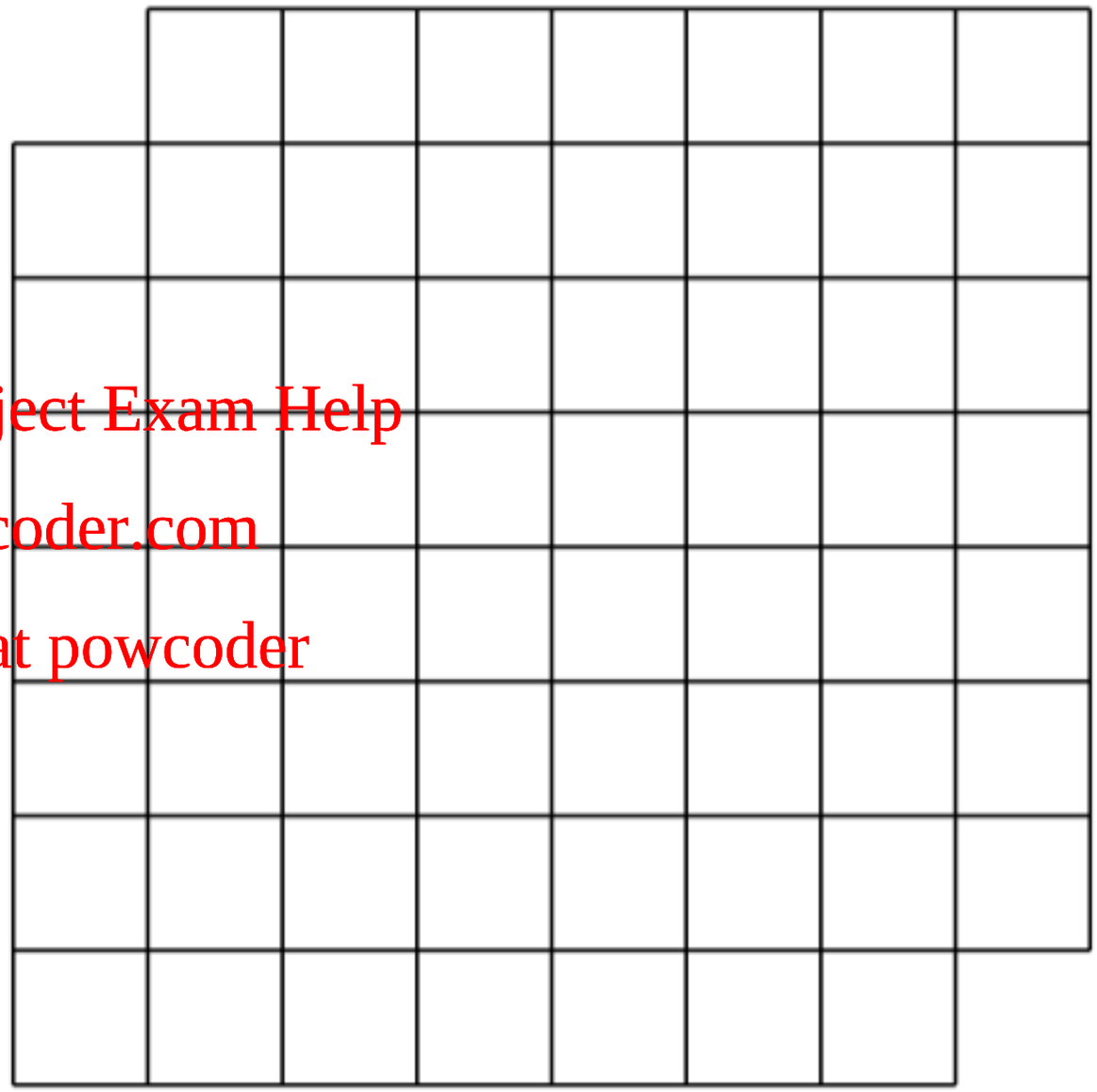
@tobycmurray

Approaching a problem

- Can we cover this board with 31 tiles of the following form?



- This is the **mutilated checkerboard problem**.

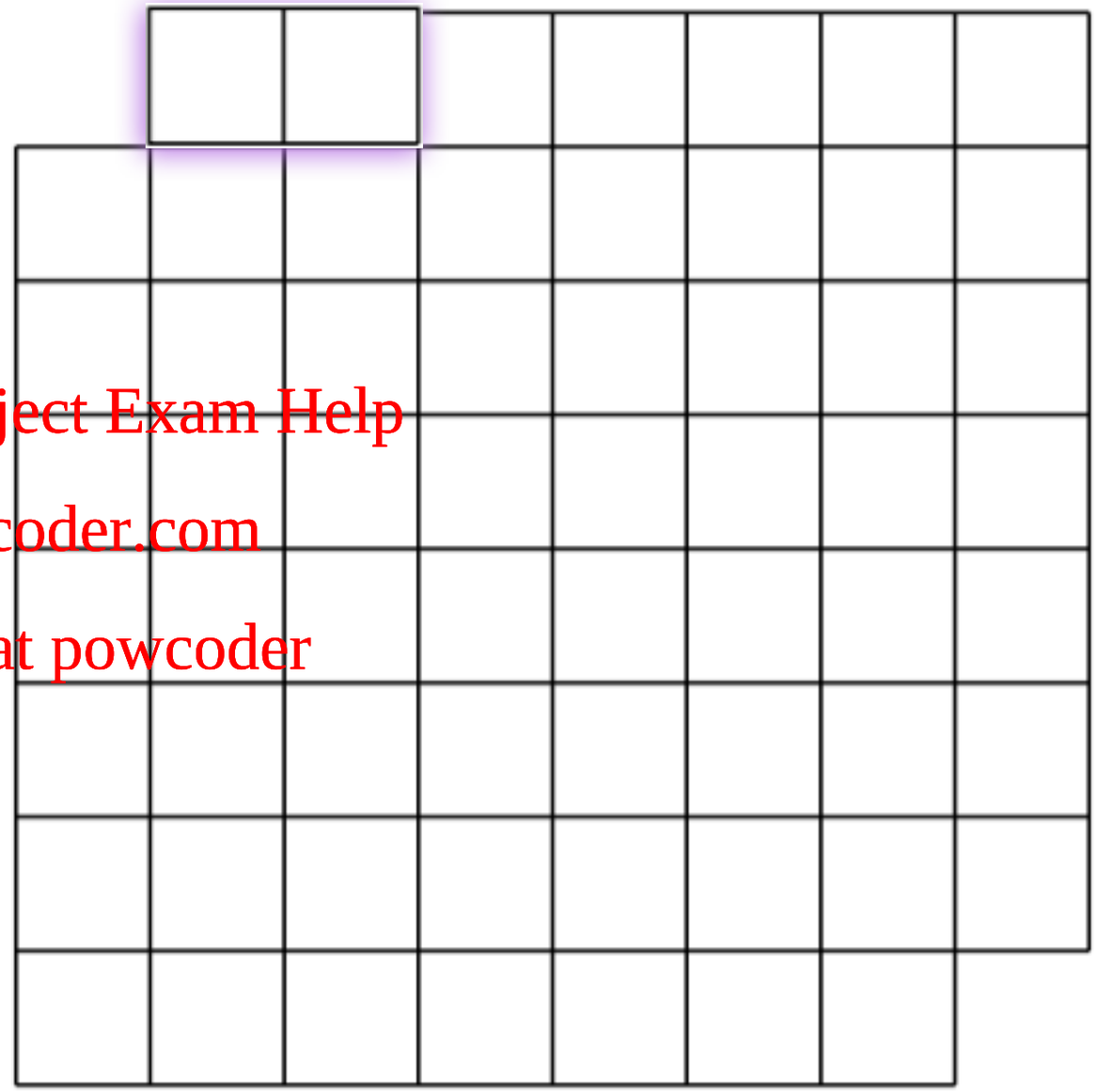


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?

- This is the **mutilated checkerboard problem**.

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

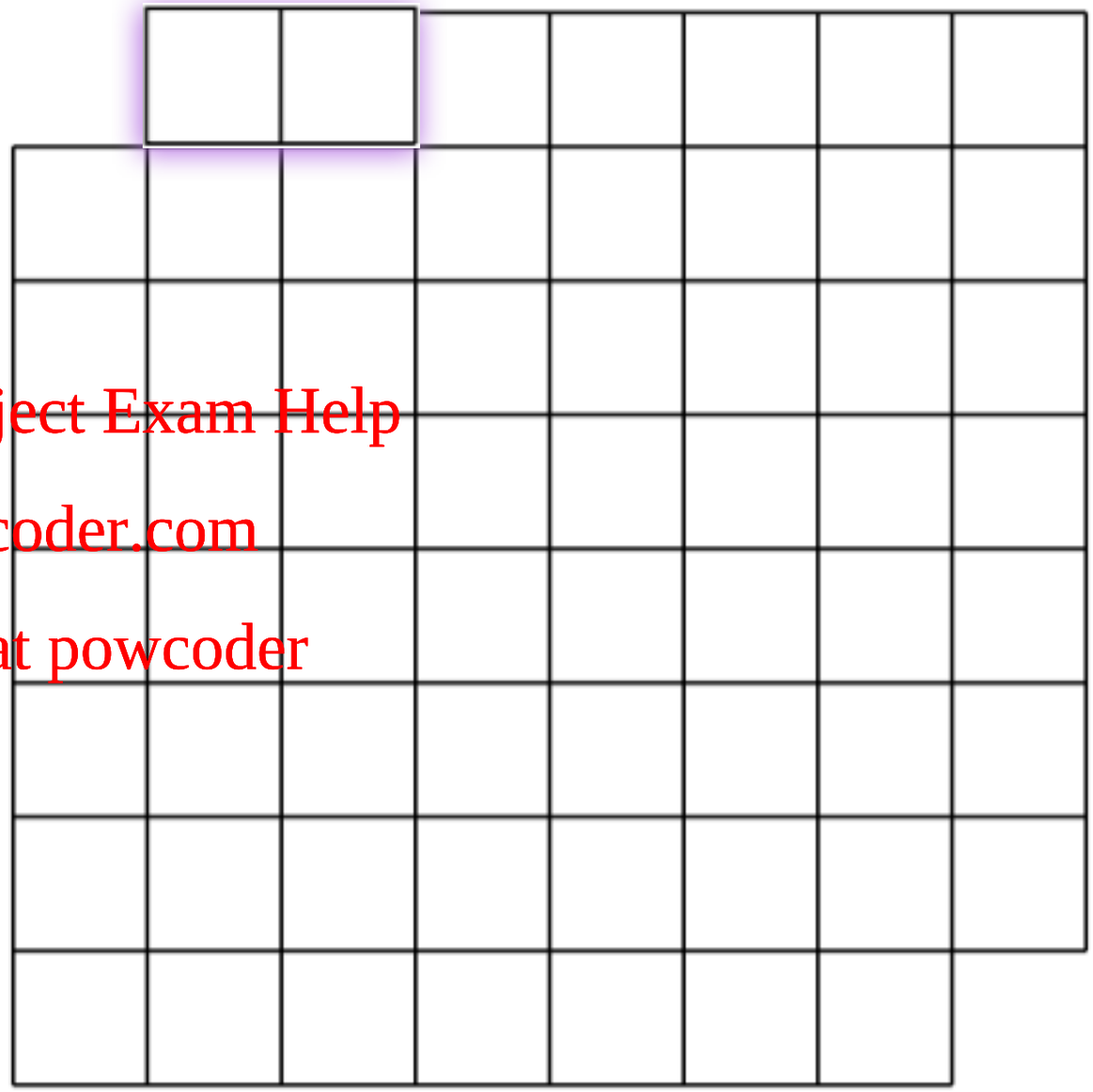
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem

- Can we cover this board with 31 tiles of the following form?

- This is the **mutilated checkerboard problem**.

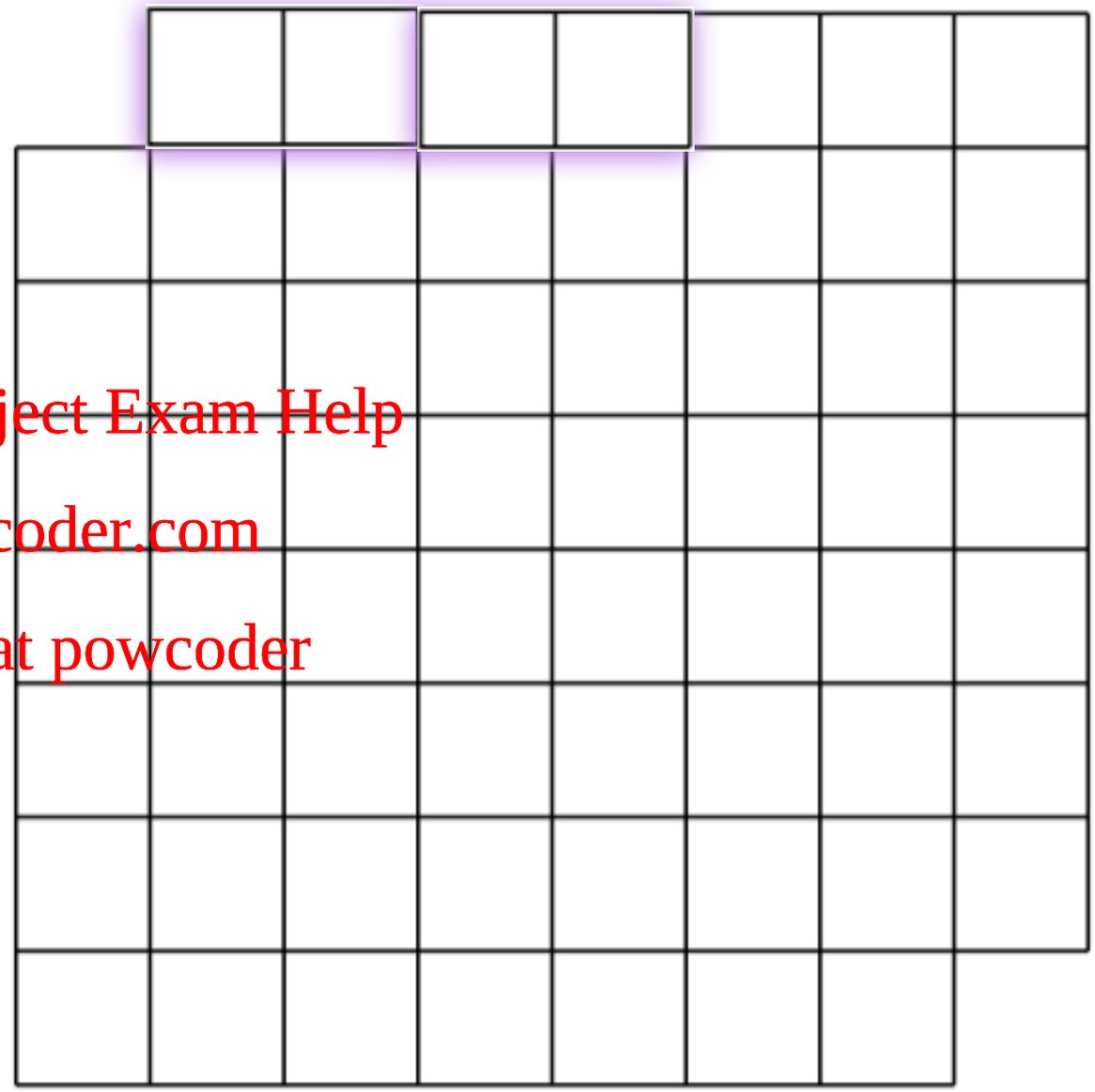


- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?

- This is the **mutilated checkerboard problem**.

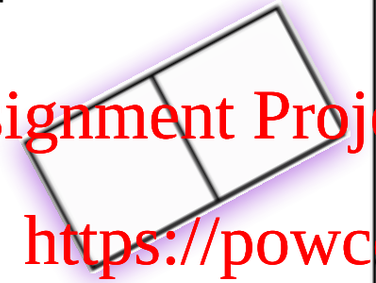
- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

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

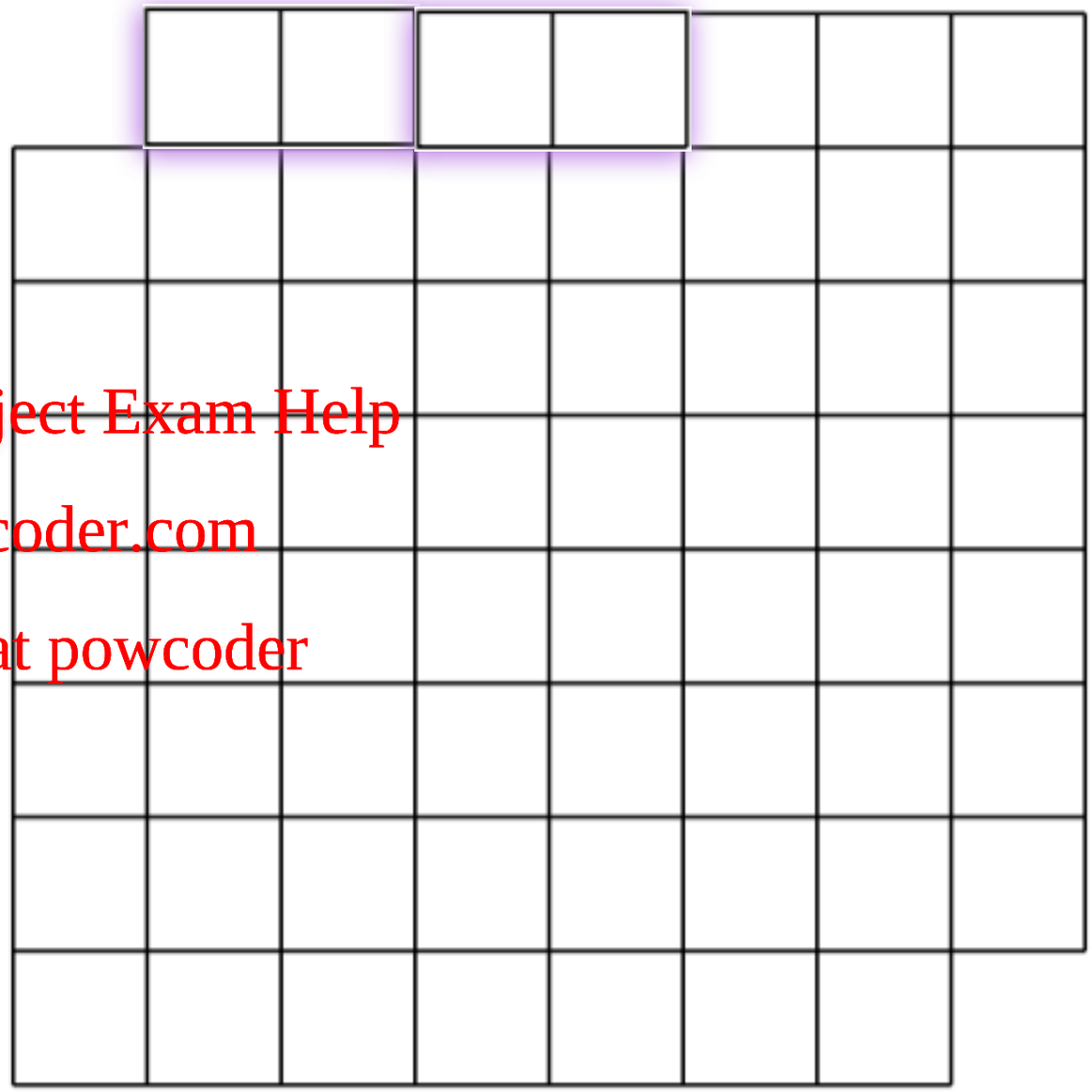
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem

- Can we cover this board with 31 tiles of the following form?



- This is the **mutilated checkerboard problem**.

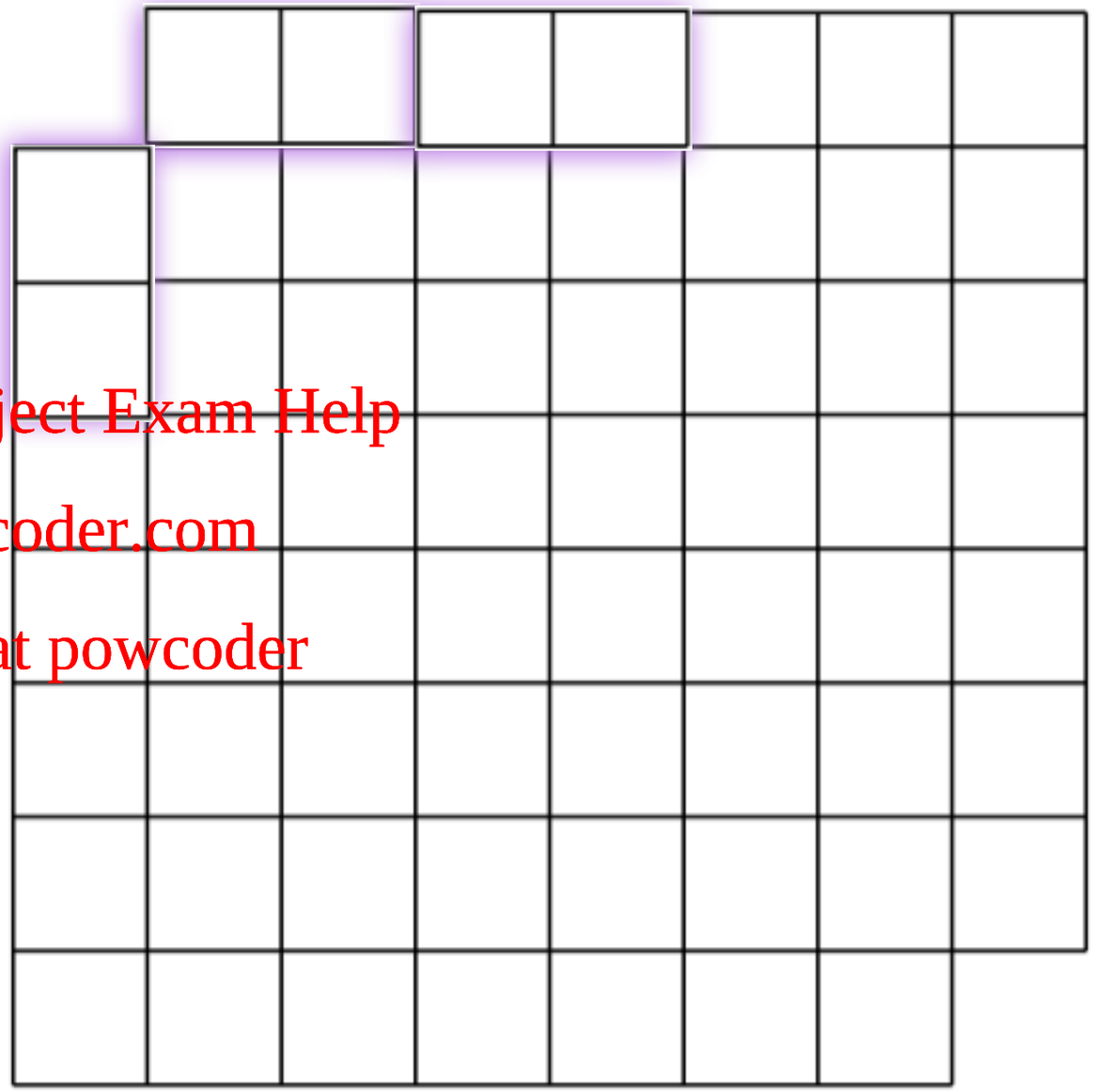


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?

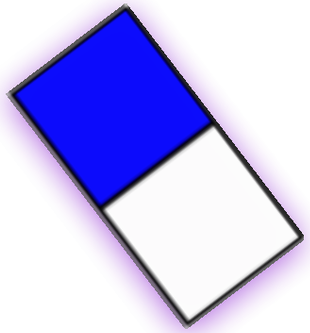
- This is the **mutilated checkerboard problem**.

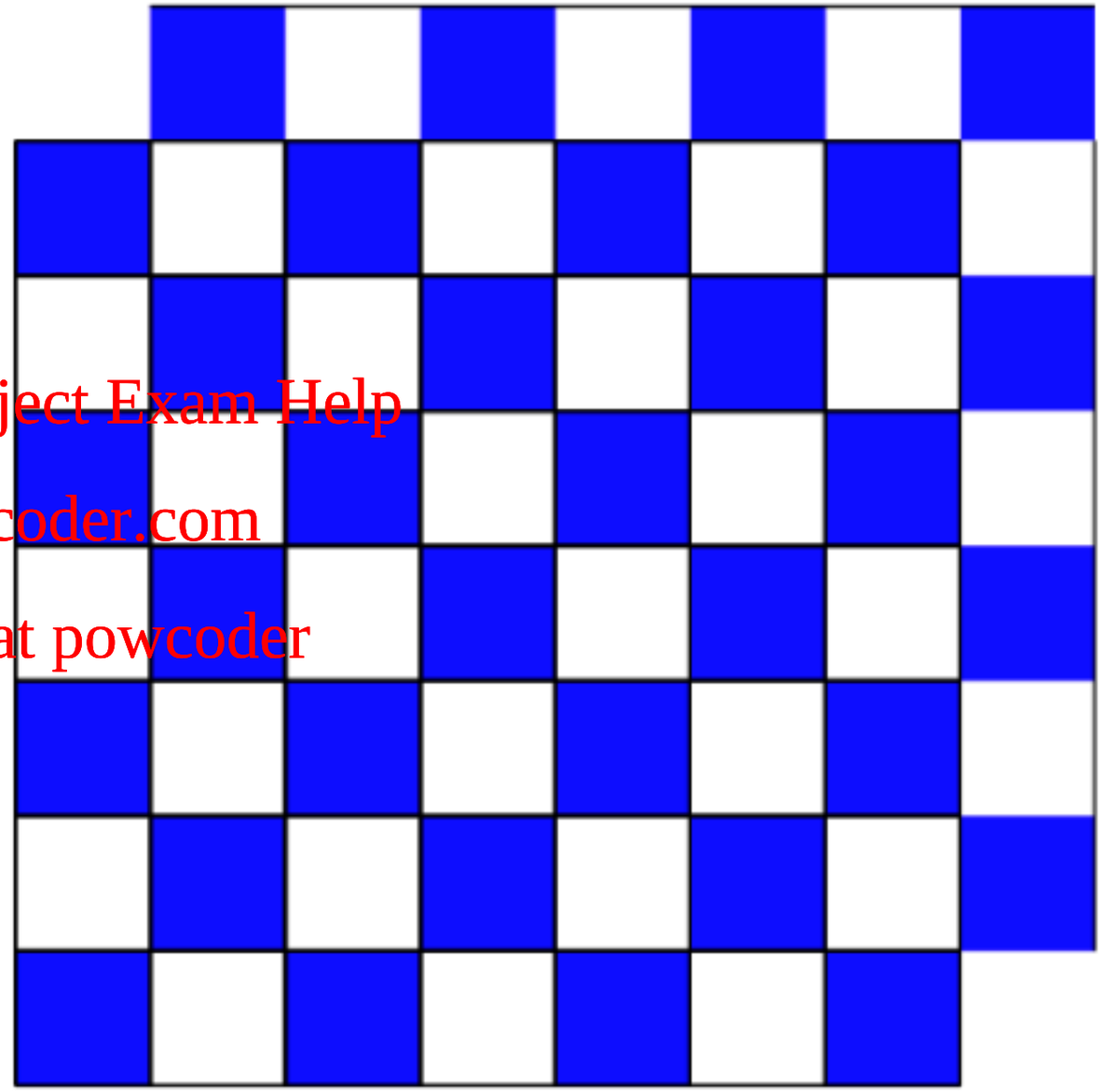
- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer? Use abstraction?



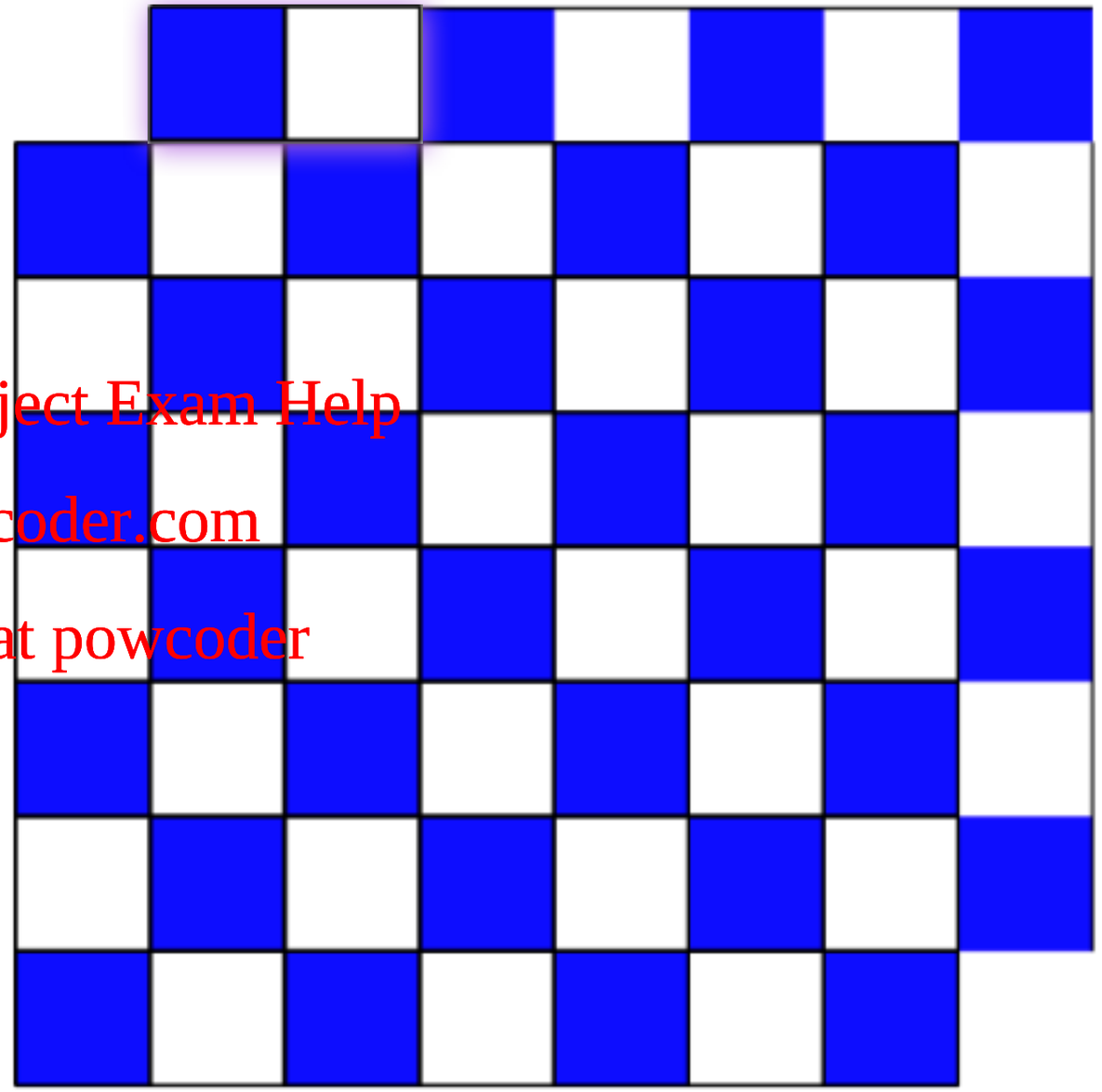


- Can we cover this board with 31 tiles of the form shown?
- Why can we quickly determine that the answer is no?
- **Hint:** Using the way the squares are coloured helps.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer? Use abstraction?

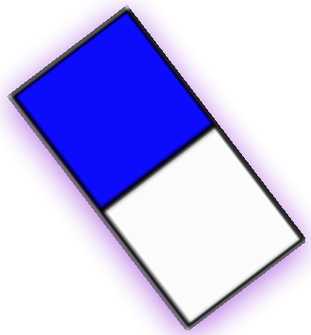


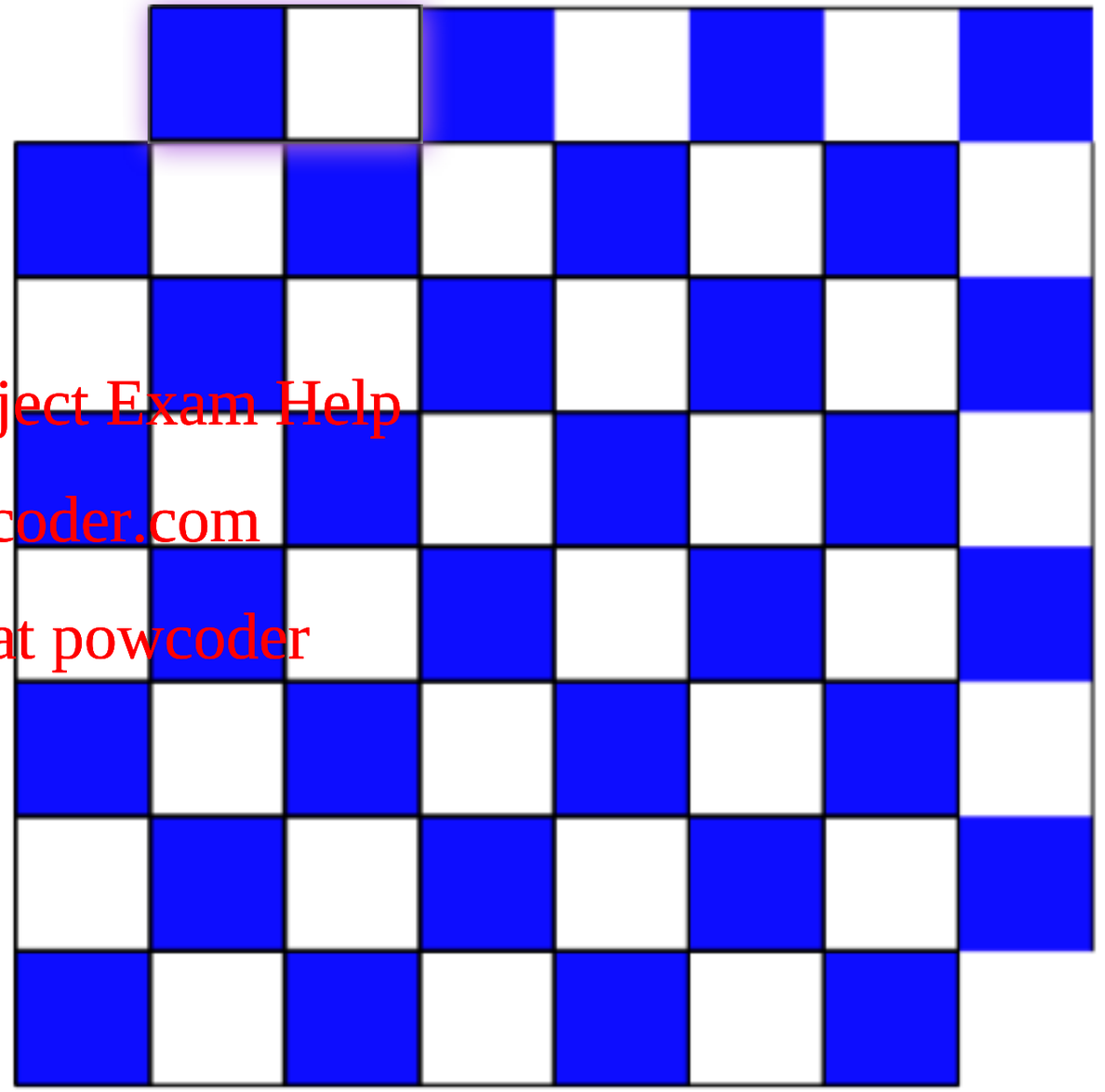
- Can we cover this board with 31 tiles of the form shown?
- Why can we quickly determine that the answer is no?
- **Hint:** Using the way the squares are coloured helps.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer? Use abstraction?



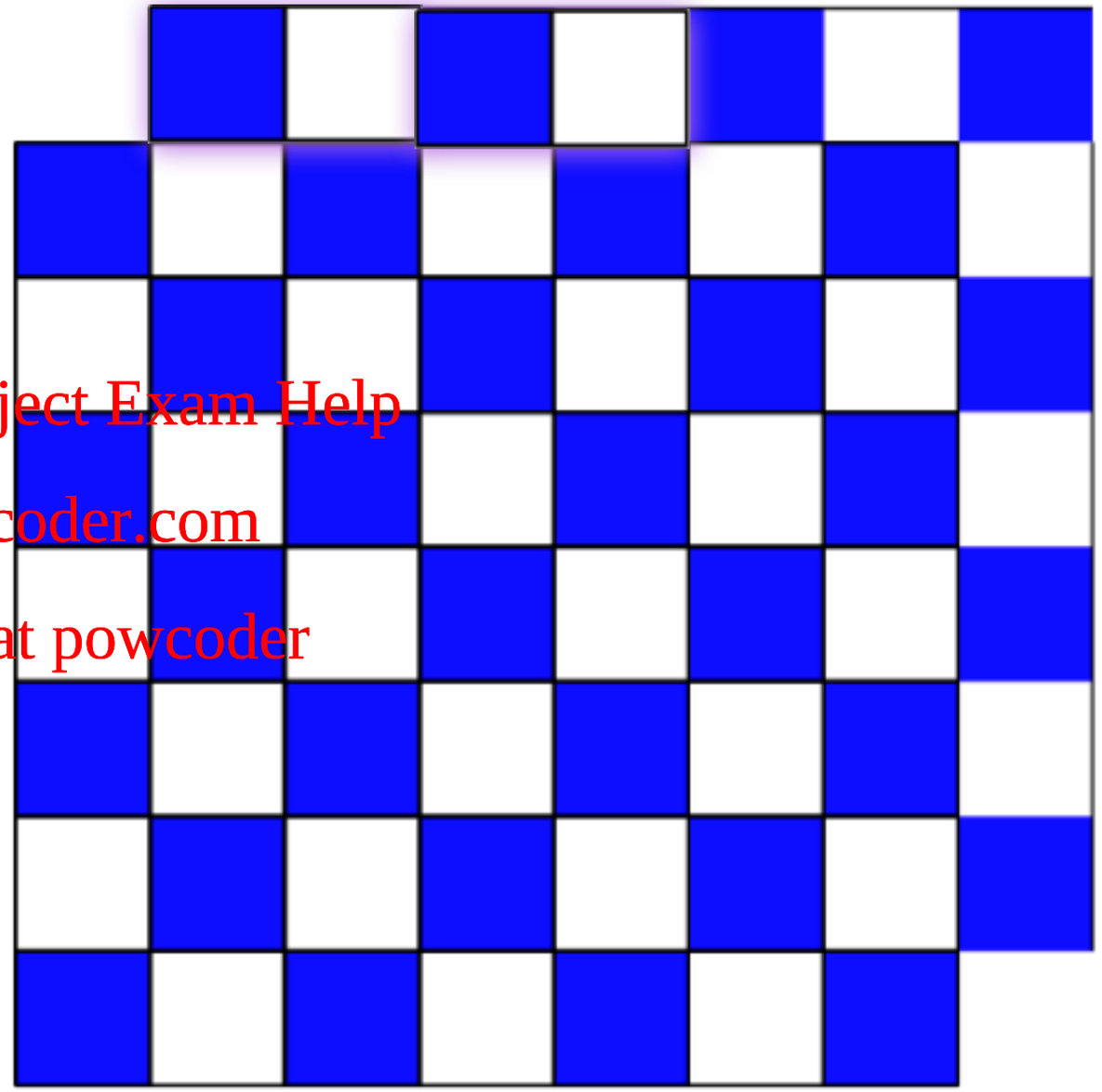


- Can we cover this board with 31 tiles of the form shown?
- Why can we quickly determine that the answer is no?
- **Hint:** Using the way the squares are coloured helps.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer? Use abstraction?



- Can we cover this board with 31 tiles of the form shown?
- Why can we quickly determine that the answer is no?
- **Hint:** Using the way the squares are coloured helps.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Algorithms and Data Structures

THE UNIVERSITY OF MELBOURNE

- **Algorithms**: for solving problems, transforming data.
- **Data structures**: for storing data, arranging data in a way that suits an algorithm.
  - **Linear** data structures: stacks and queues
  - Trees and graphs
  - Dictionaries
- Which data structures are you familiar with?

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Exercise

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

- Pick you favourite data structure and describe
  - How to insert and item into the data structure
  - How to find an item
  - How to handle duplicate items

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

THE UNIVERSITY OF MELBOURNE

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.
- The downside of an array is that maintaining a contiguous bank of cells with information can be difficult and time-consuming.

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

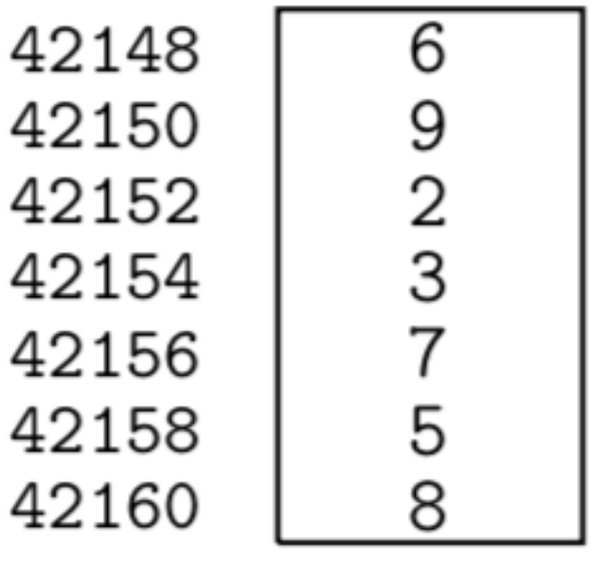
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

THE UNIVERSITY OF MELBOURNE

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.
- The downside of an array is that maintaining a contiguous bank of cells with information can be difficult and time-consuming.

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



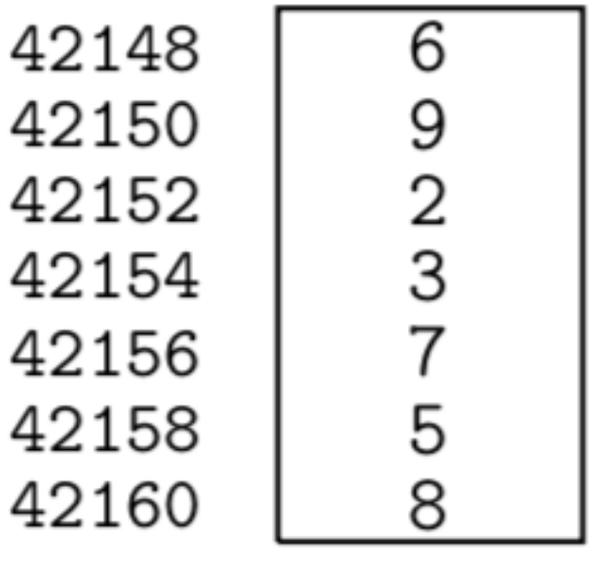
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.
- The downside of an array is that maintaining a contiguous bank of cells with information can be difficult and time-consuming.

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

How many bytes does each integer occupy here?



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

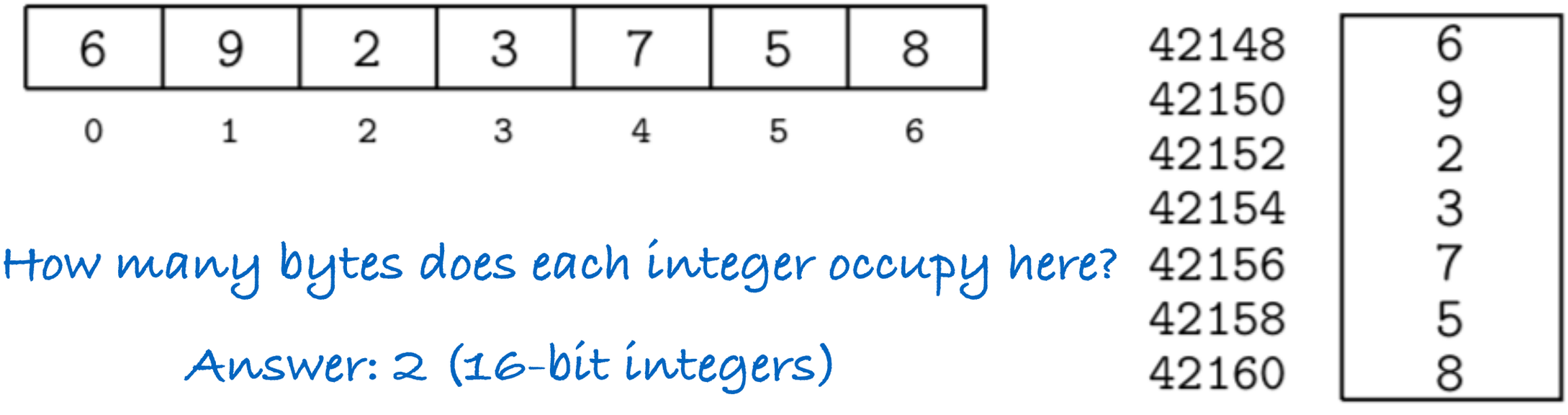
Primitive Data Structures: The Array

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.

Assignment Project Exam Help
https://powcoder.com

- The downside of an array is that maintaining a contiguous bank of cells with information can be difficult and time-consuming.

Add WeChat powcoder

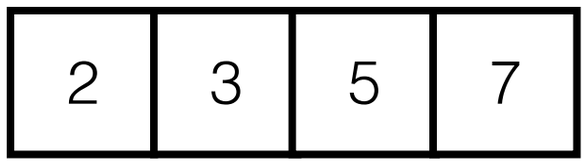


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

THE UNIVERSITY OF MELBOURNE

An array **x**:



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

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

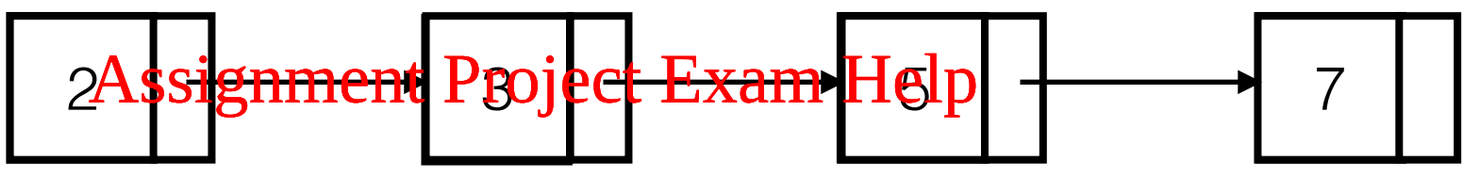
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

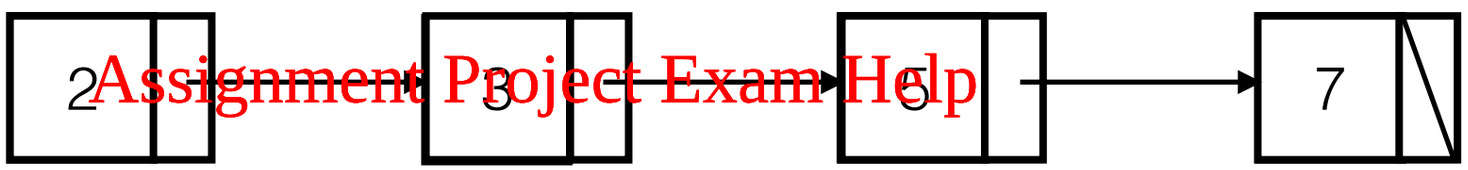
Primitive Data Structures: The Linked List



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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

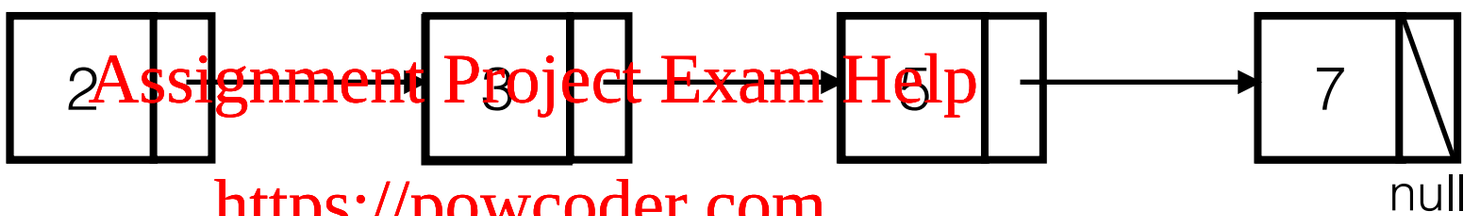
Primitive Data Structures: The Linked List



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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

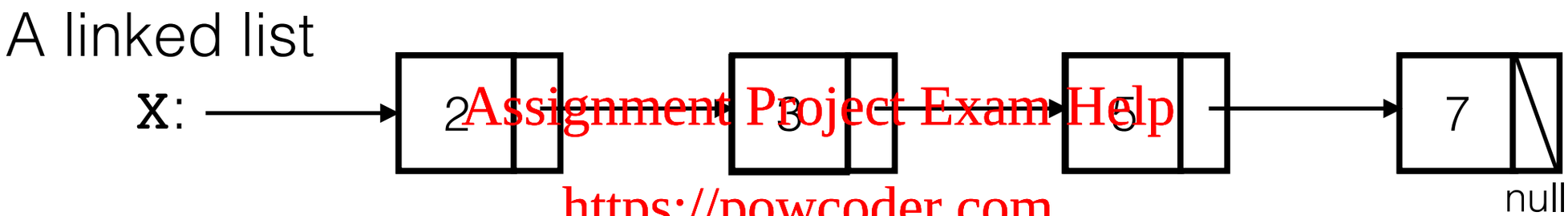
Primitive Data Structures: The Linked List



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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

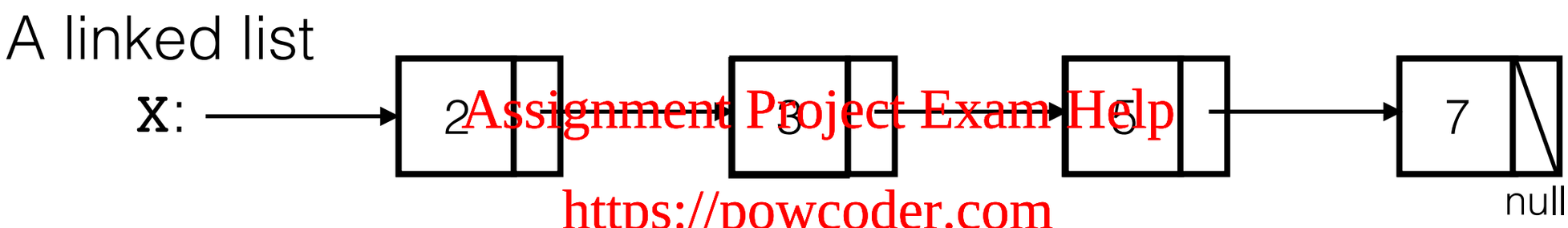
Primitive Data Structures: The Linked List



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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List



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

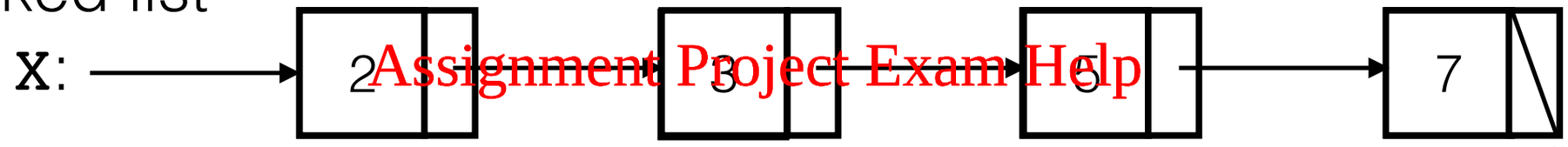
Suppose variable `x` holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

THE UNIVERSITY OF MELBOURNE

A linked list



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

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

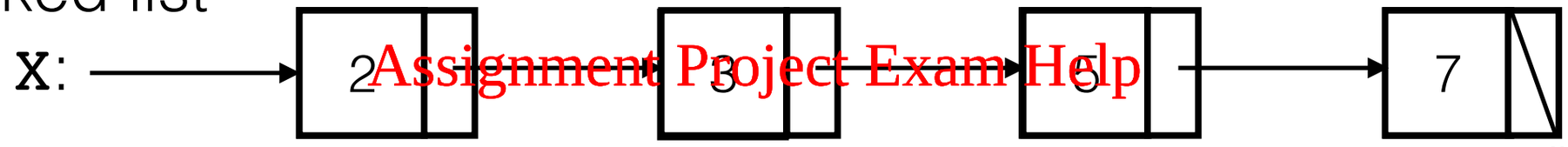
42148	3
42150	42152
42152	5
42154	42164
42156	
42158	
42160	2
42162	42148
42164	7
42166	0

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

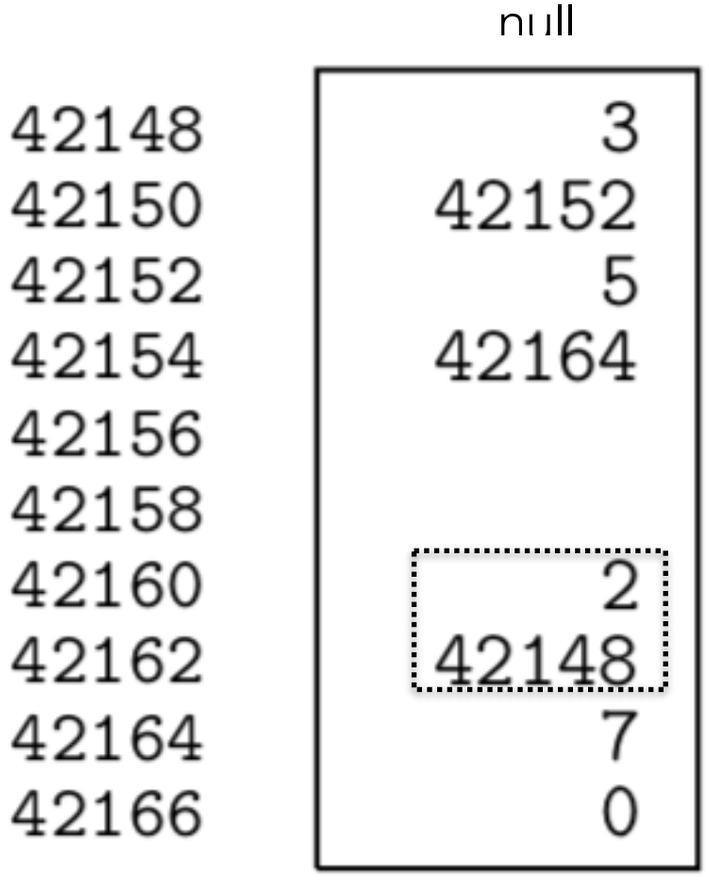
THE UNIVERSITY OF MELBOURNE

A linked list



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

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

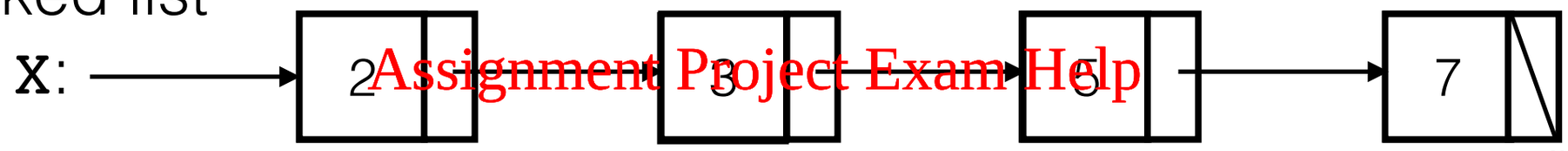


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

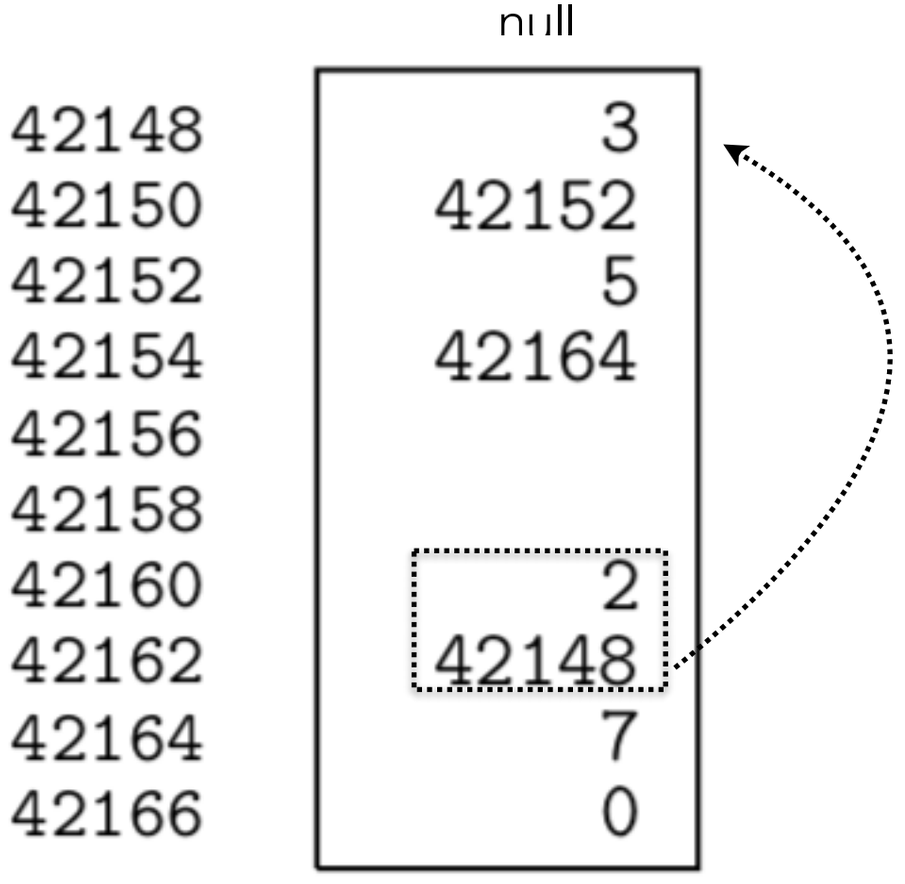
THE UNIVERSITY OF MELBOURNE

A linked list



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

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

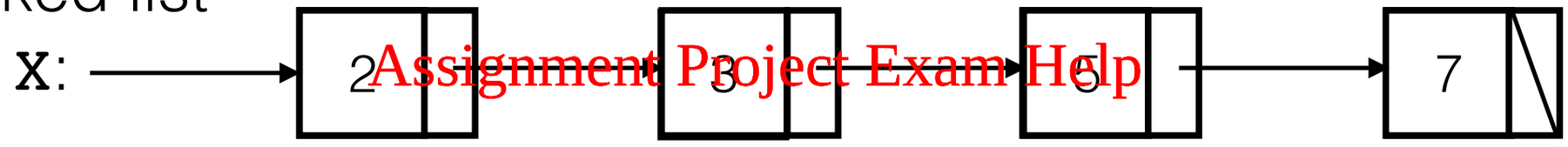


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

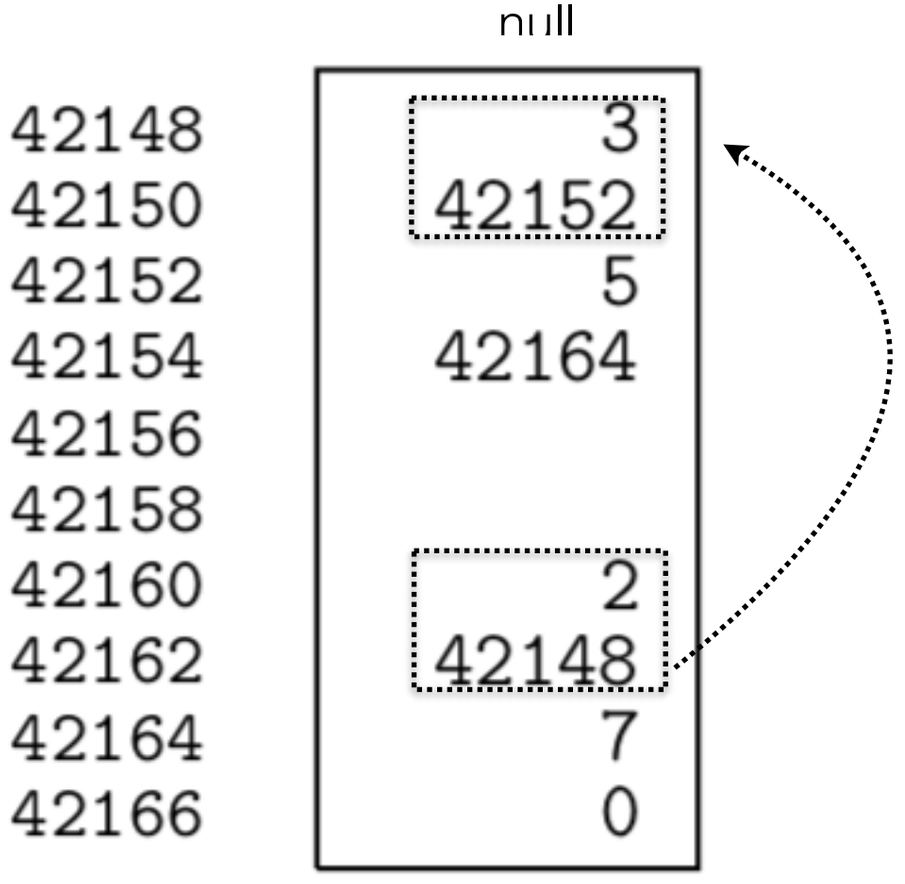
THE UNIVERSITY OF MELBOURNE

A linked list



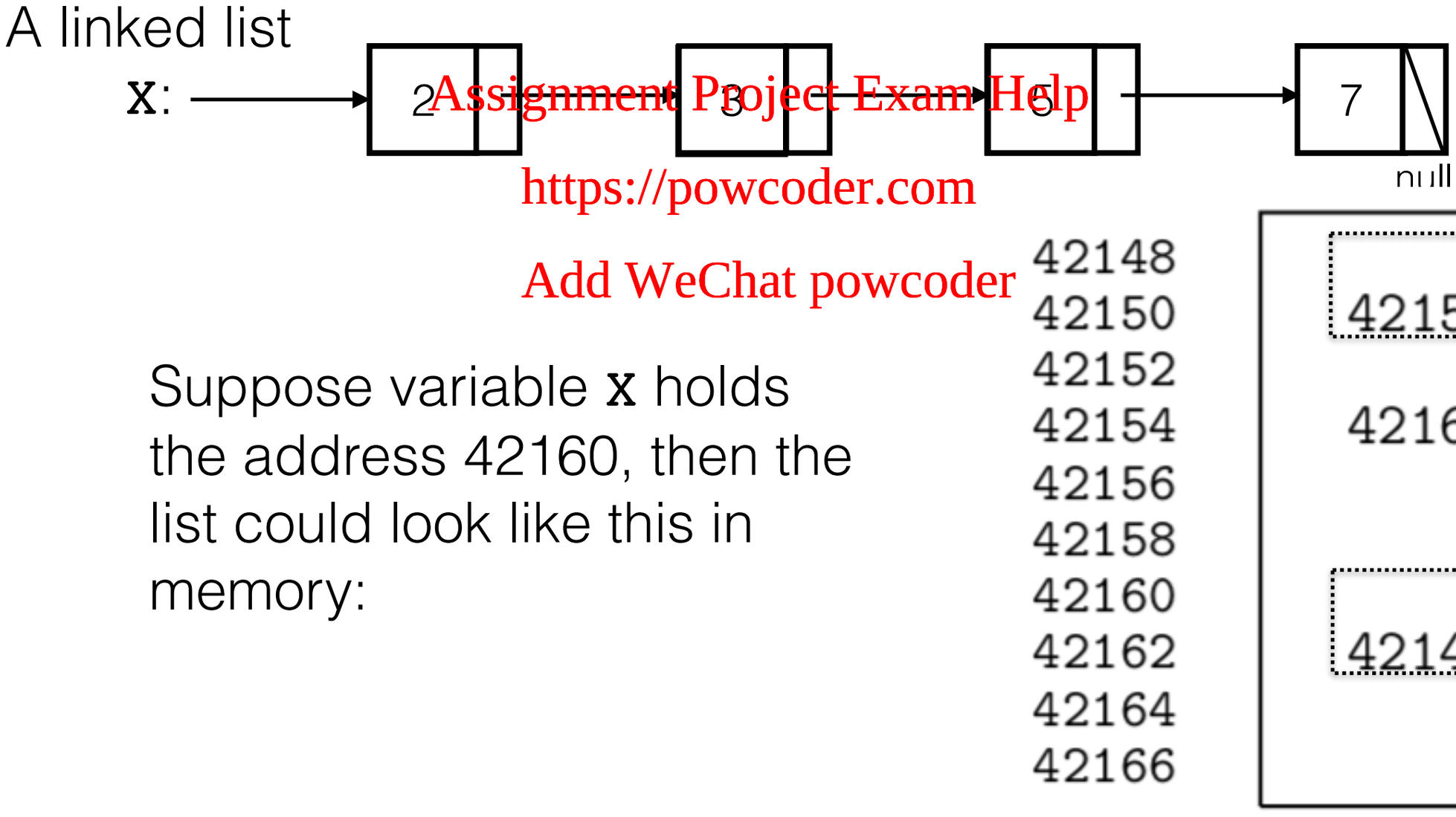
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List



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

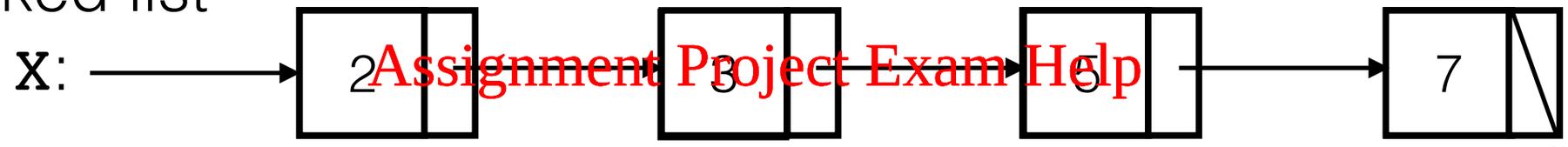
Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

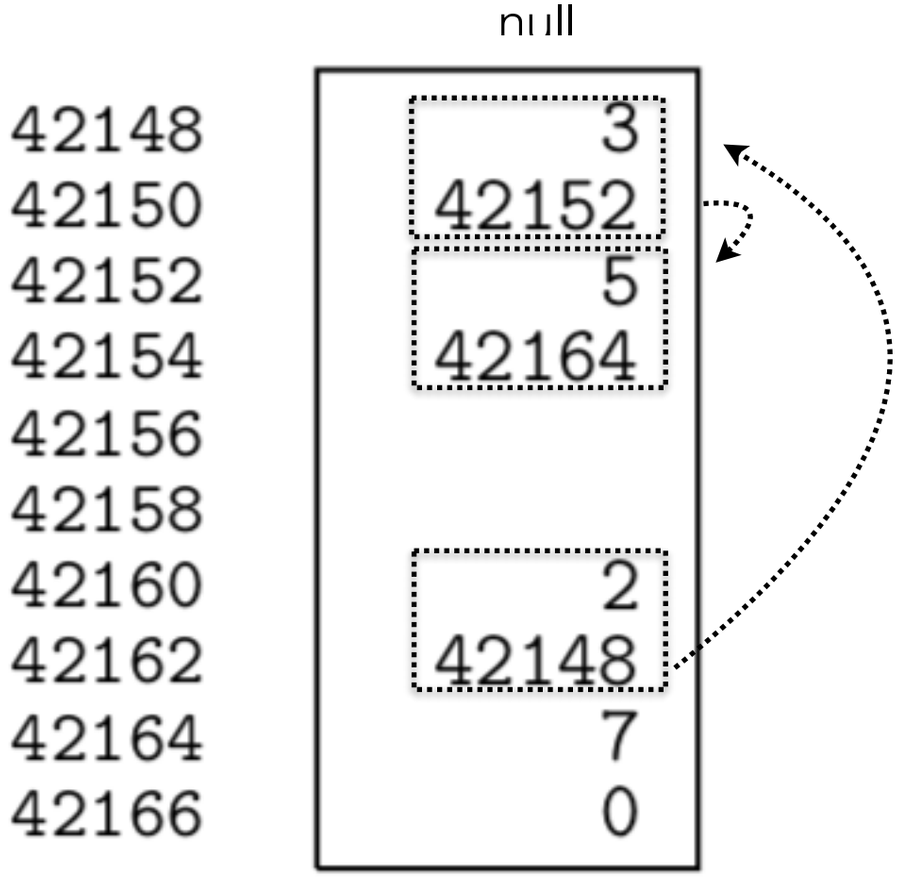
THE UNIVERSITY OF MELBOURNE

A linked list



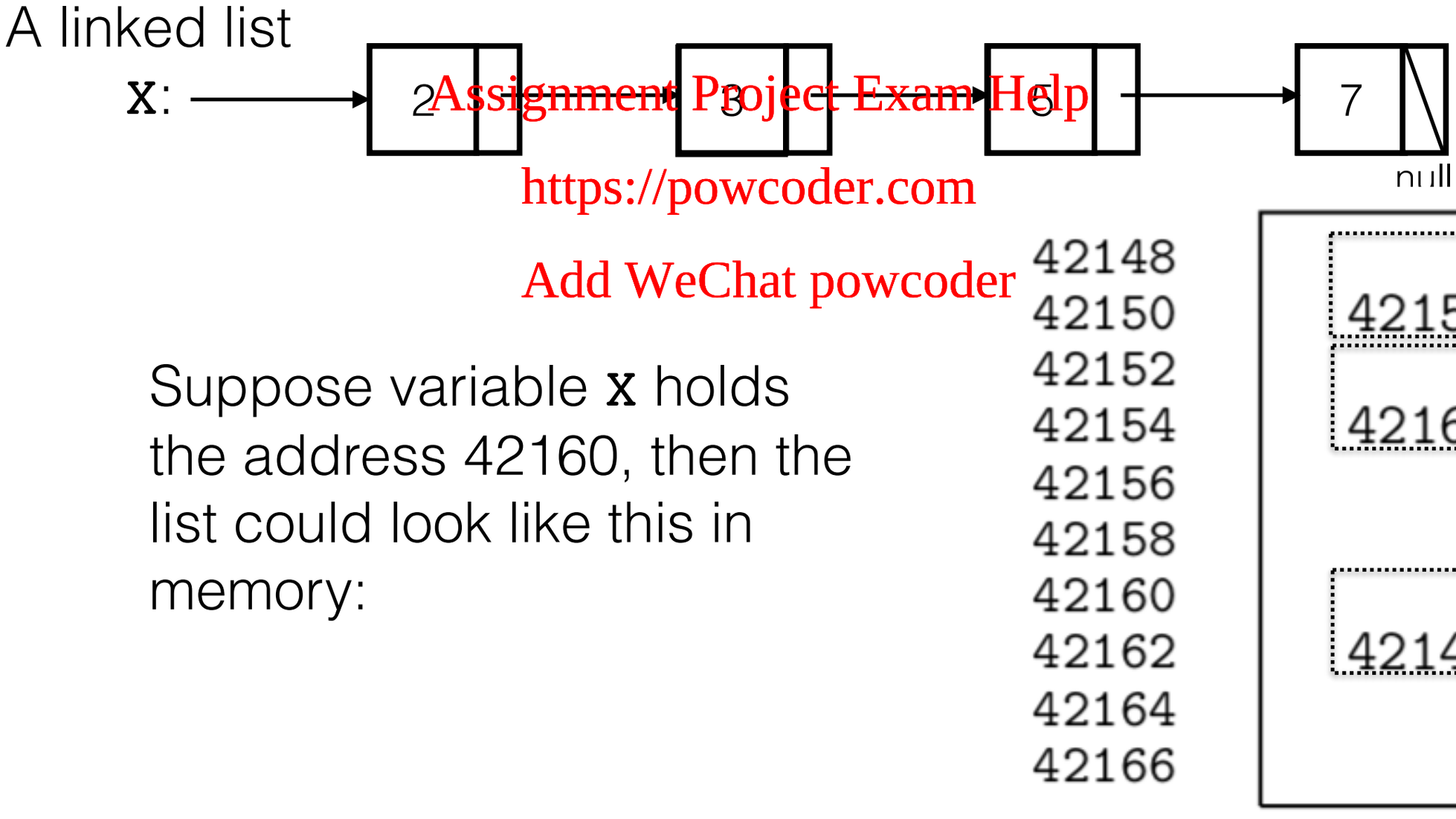
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

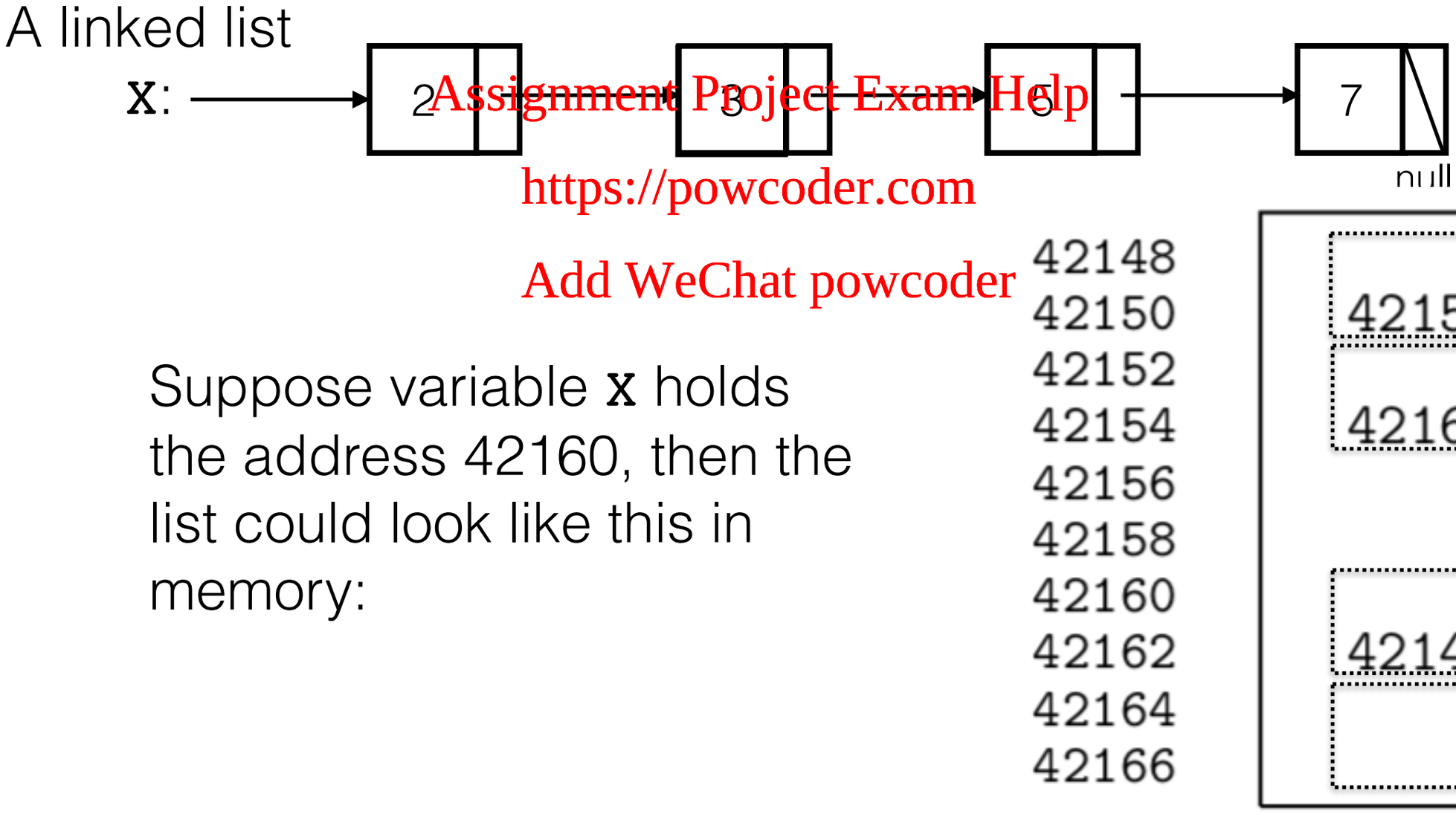


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List



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

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

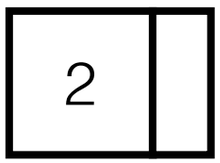
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



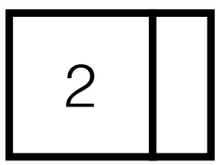
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



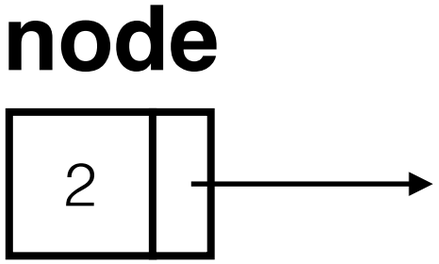
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



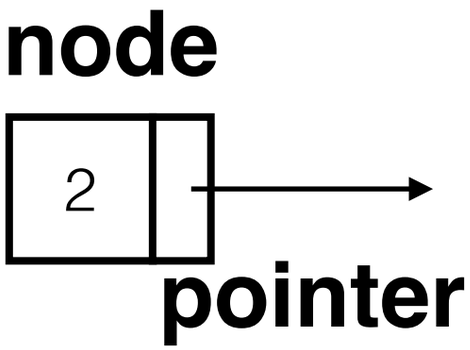
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



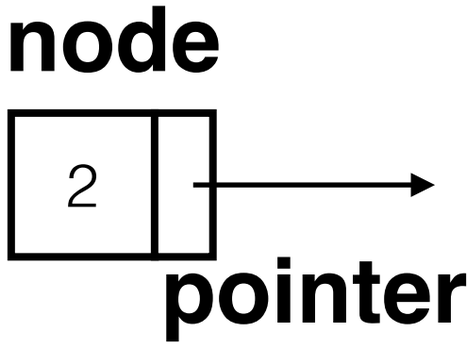
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

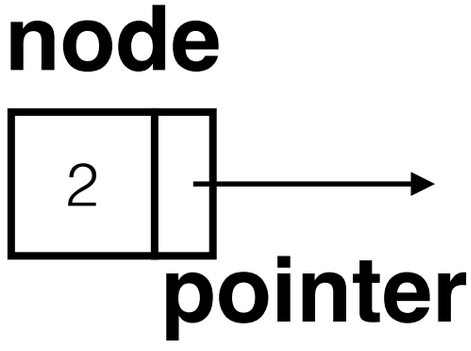
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

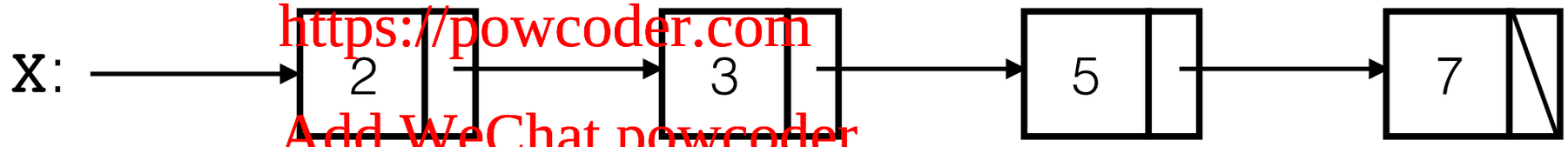
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

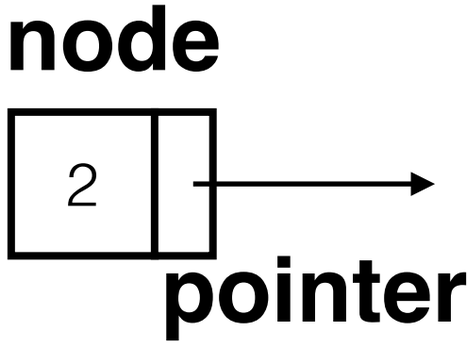
Assignment Project Exam Help



https://powcoder.com
Add WeChat powcoder

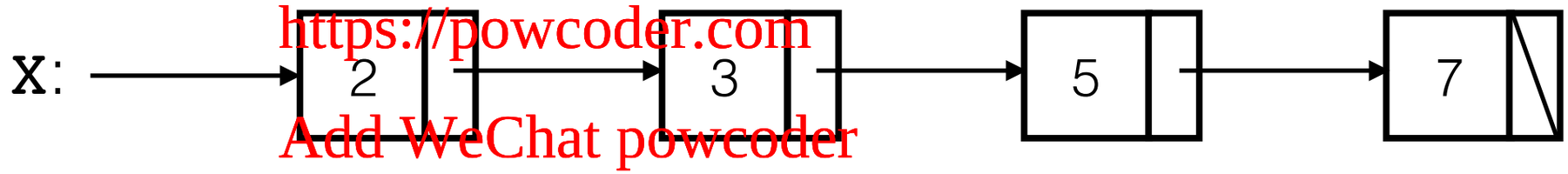
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

Assignment Project Exam Help

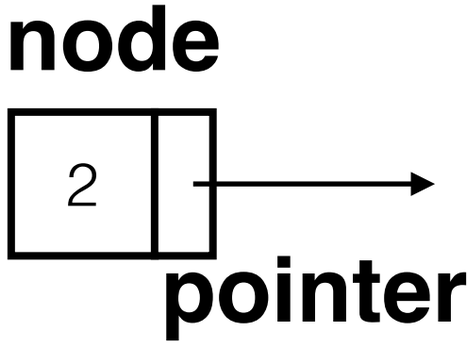


https://powcoder.com
Add WeChat powcoder

`x` is (a pointer to) the **head node** of the list

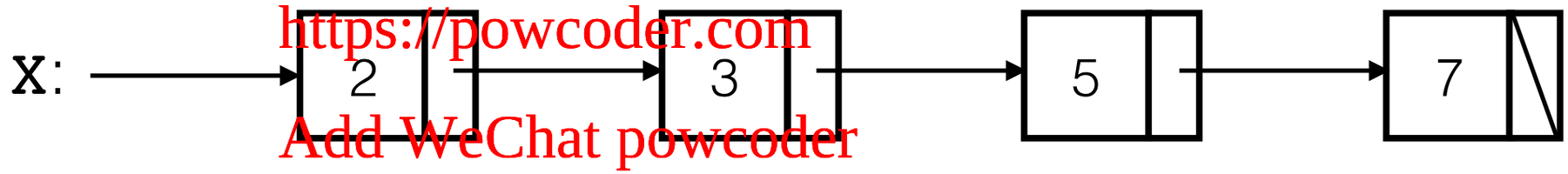
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



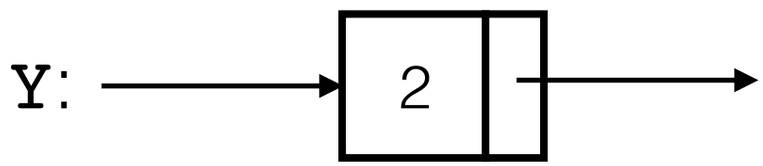
(in Java: “reference”)

Assignment Project Exam Help



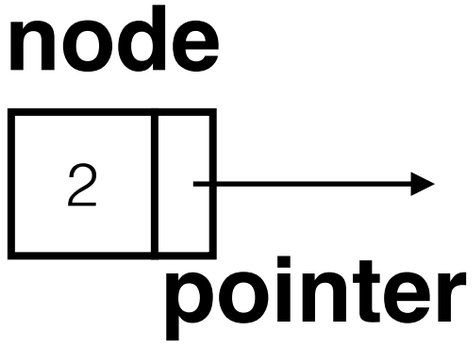
https://powcoder.com
Add WeChat powcoder

`x` is (a pointer to) the **head node** of the list



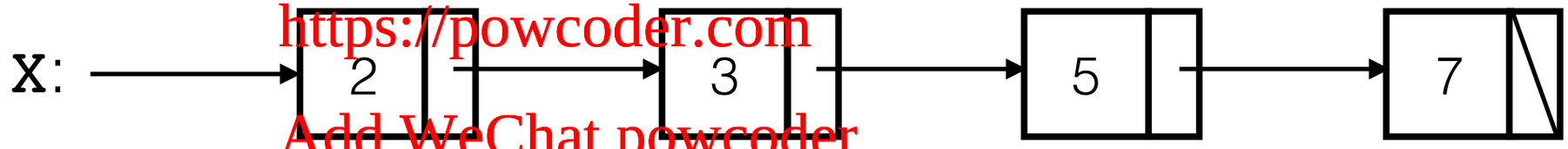
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

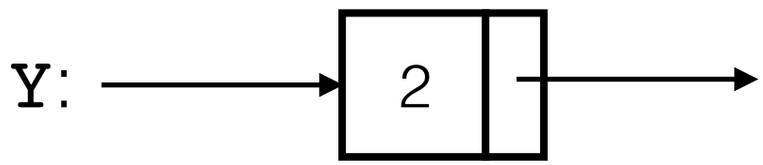


(in Java: “reference”)

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



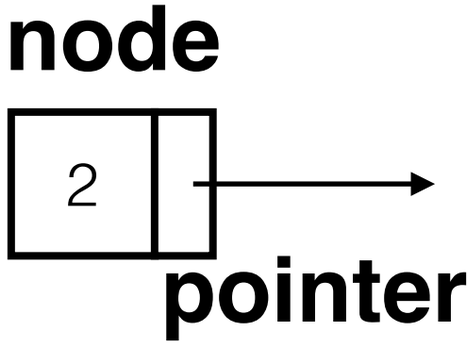
`x` is (a pointer to) the **head node** of the list



“`Y.val`” *refers to*

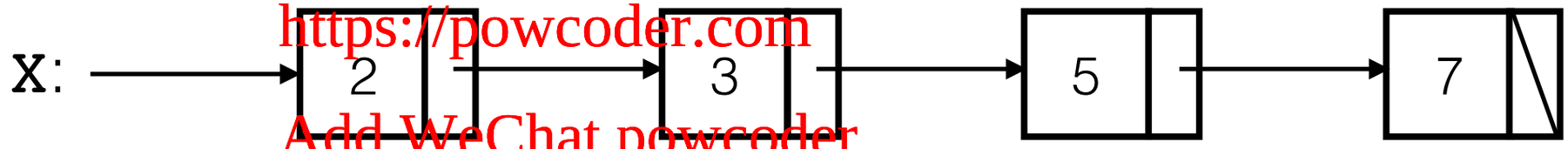
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

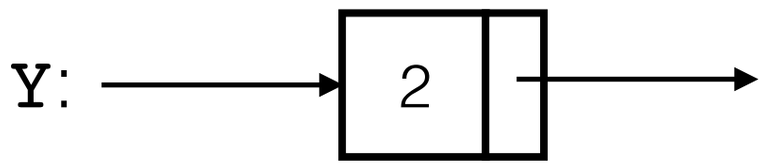


(in Java: “reference”)

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



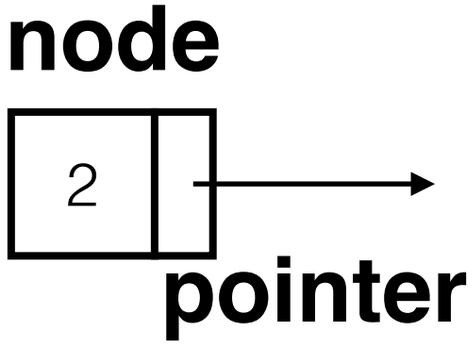
`x` is (a pointer to) the **head node** of the list



“`y.val`” *refers to*

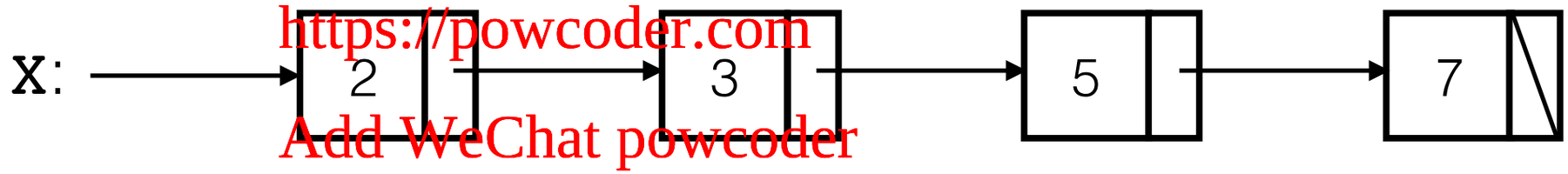
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



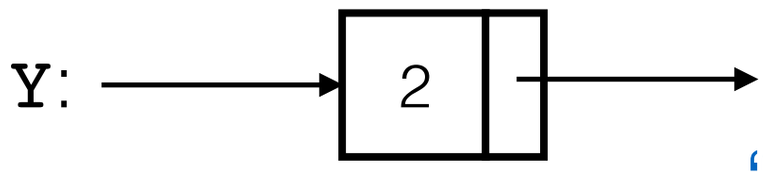
(in Java: “reference”)

Assignment Project Exam Help



https://powcoder.com
Add WeChat powcoder

`x` is (a pointer to) the **head node** of the list

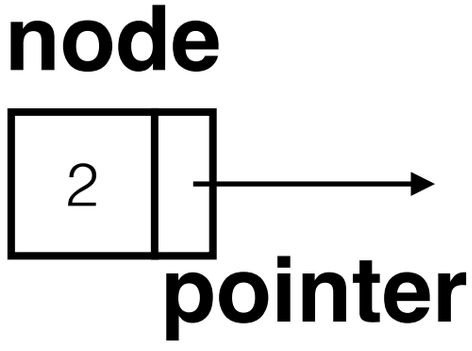


“`Y.val`” refers to

“`Y.next`” refers to

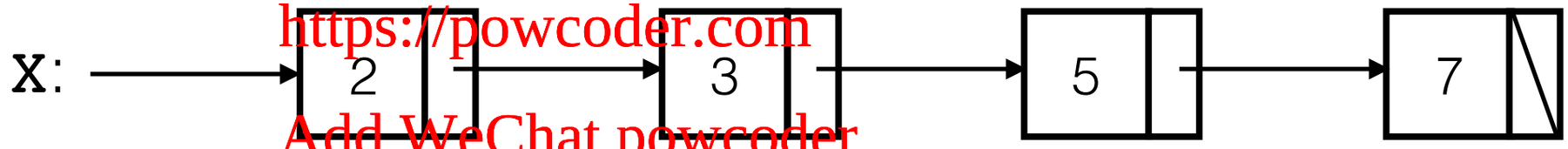
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

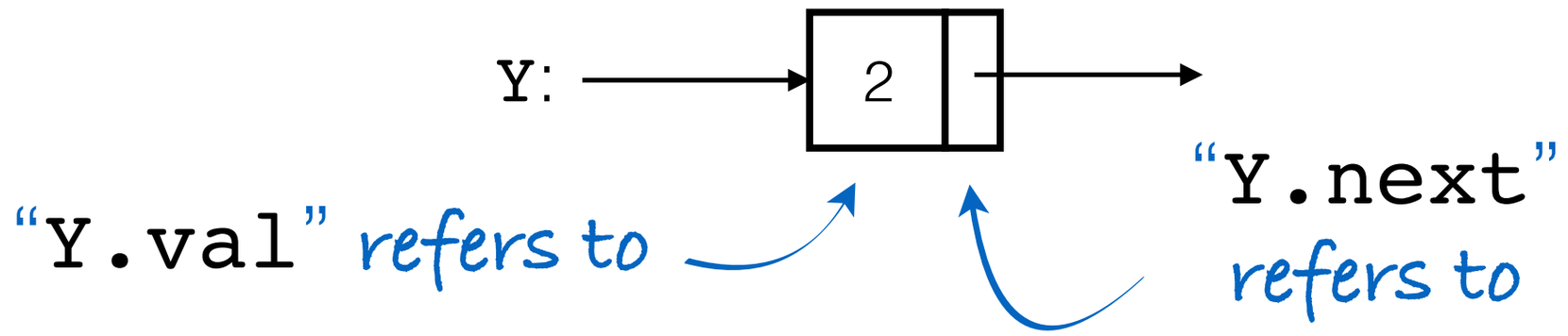


(in Java: “reference”)

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



`x` is (a pointer to) the **head node** of the list



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Linked List

- Often we use a dummy head node that points to the first object, or to a special `null` object that represents an empty list. This makes it easier to write functions that insert or delete elements.

Assignment Project Exam Help
https://powcoder.com

- Inserting and deleting elements is very fast: just move a few links around.

Add WeChat powcoder

- Finding the *i*th element can be time-consuming.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of **find(Y,7,6)**

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

Let's trace the execution of **find(Y,7,6)**

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

- Walk through the array (of length n)
- For example, to locate item **x**.

A: Y x: 7

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y x: 7 n: 6

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

- Walk through the array (of length n)
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 0

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of **find(Y,7,6)**

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

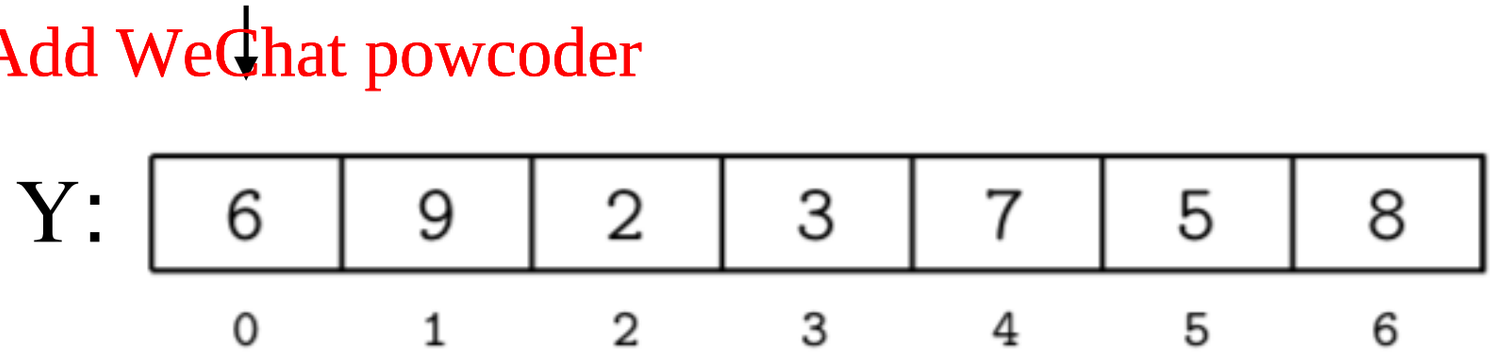
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 0

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

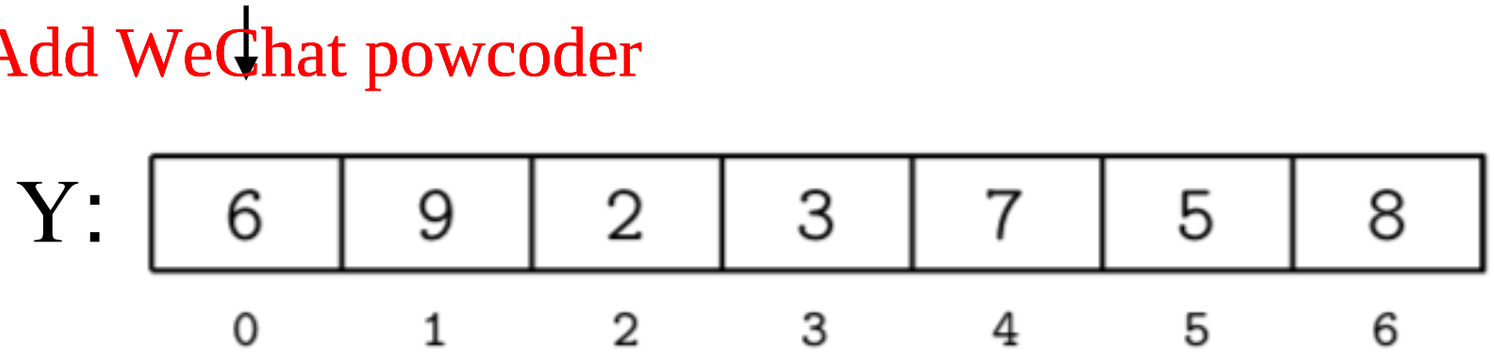
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 1

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

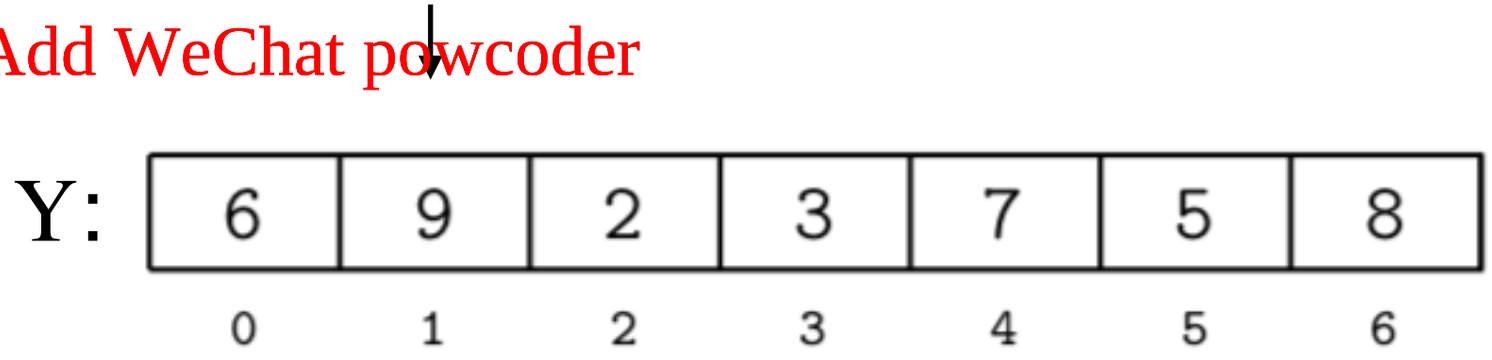
- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 1

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

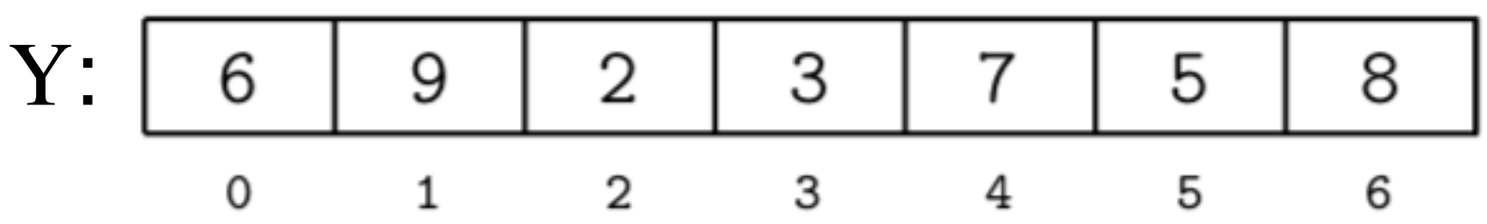
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 2

```
function find(A,x,n)
    j ← 0
while j < n
    if A[j] = x
        return j
    j ← j+1
return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

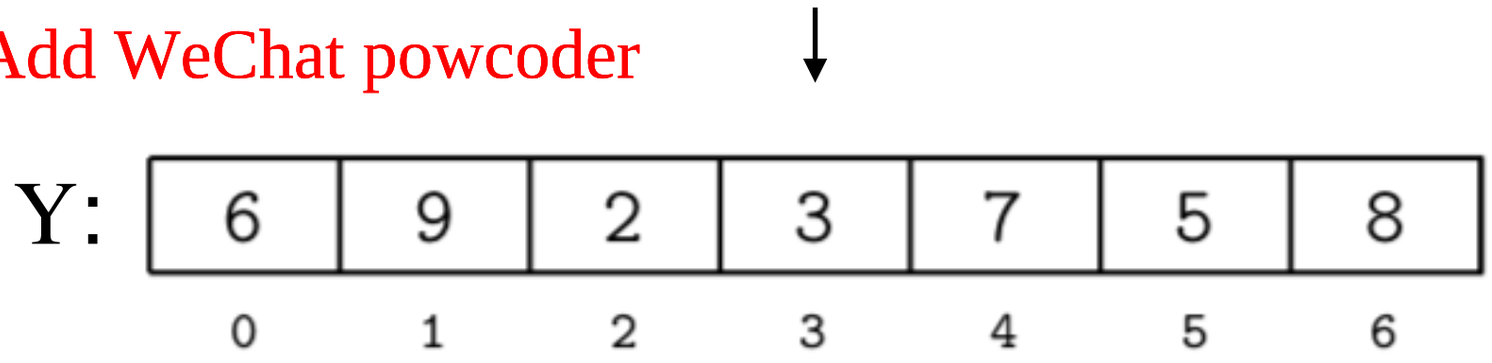
- Walk through the array (of length n)
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 3

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

THE UNIVERSITY OF MELBOURNE

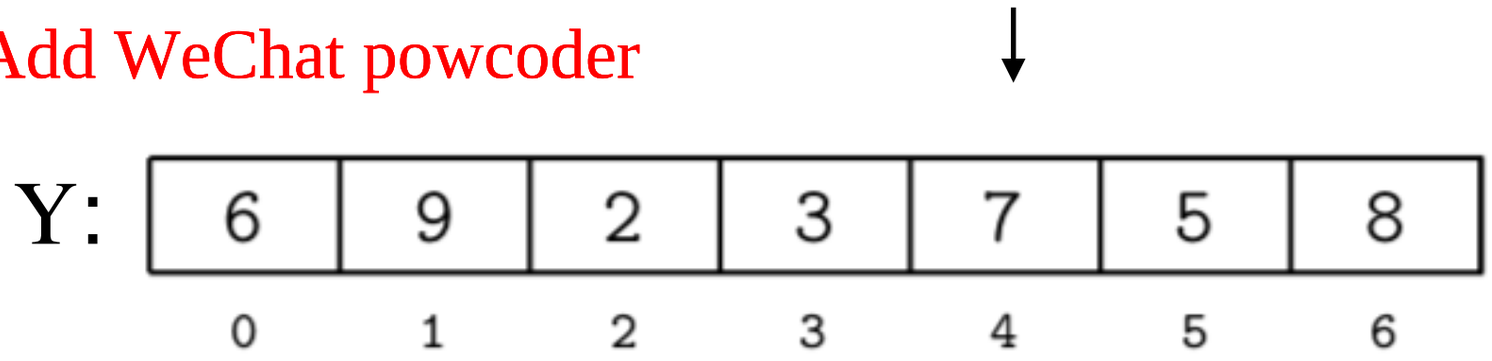
- Walk through the array (of length n)
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 4

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

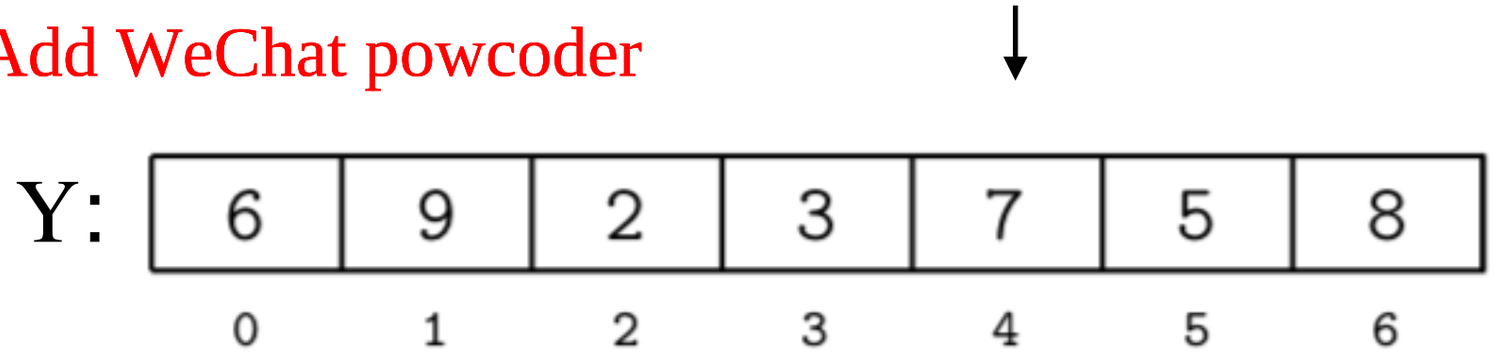
- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 4

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

A[j]



Let's trace the execution of find(Y,7,6)

(returns 4)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item `x`.

```
function find(head,x)
    p ← head
  while p ≠ null
     if p.val = x
        return p
    p ← p.next
return null
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

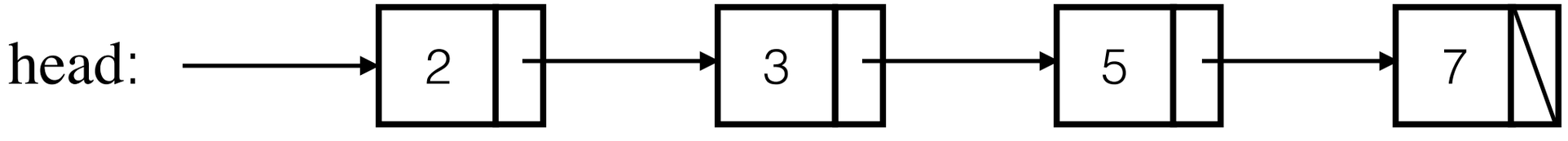
- For example, to locate item `x`.

```
function find(head,x)
    p ← head
  while p ≠ null
     if p.val = x
        return p
    p ← p.next
  return null
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

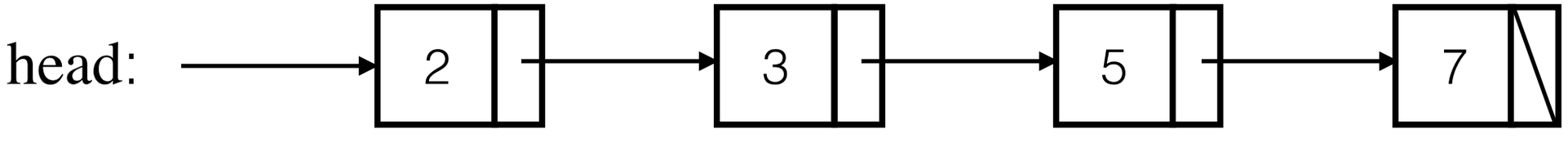
- For example, to locate item **x**.

(note similarity to array version)

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

```
function find(head,x)
    p ← head
  while p ≠ null
     if p.val = x
        return p
    p ← p.next
  return null
```

```
function find(A,x,n)
    i ← 0
  while j < n
     if A[j] = x
        return j
    j ← j+1
  return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

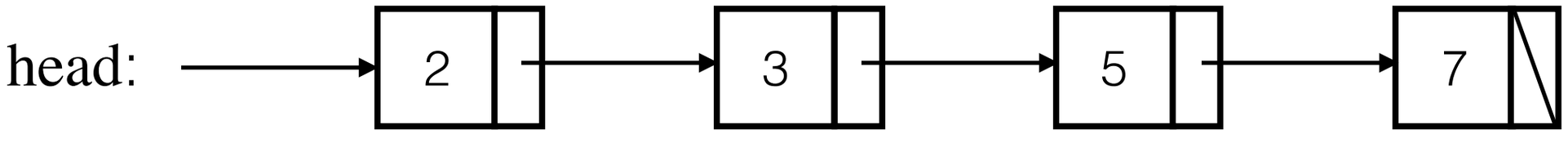
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item **x**.

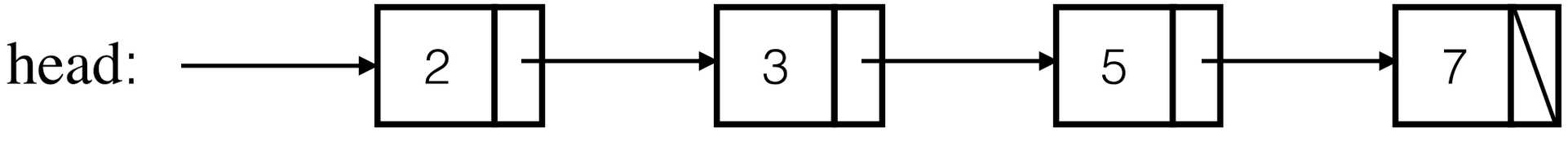
(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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

p:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

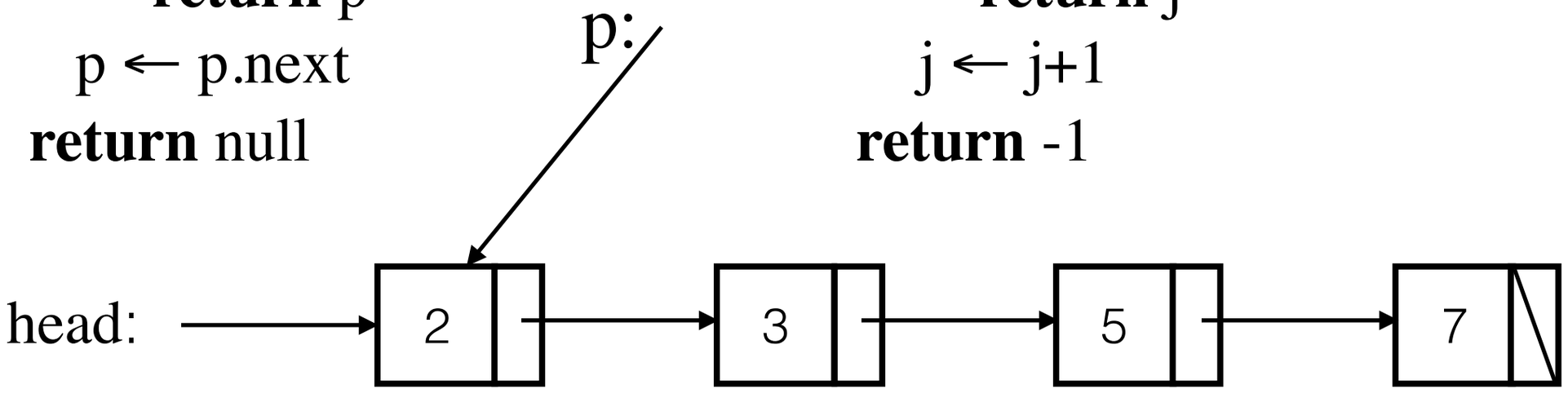
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

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



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

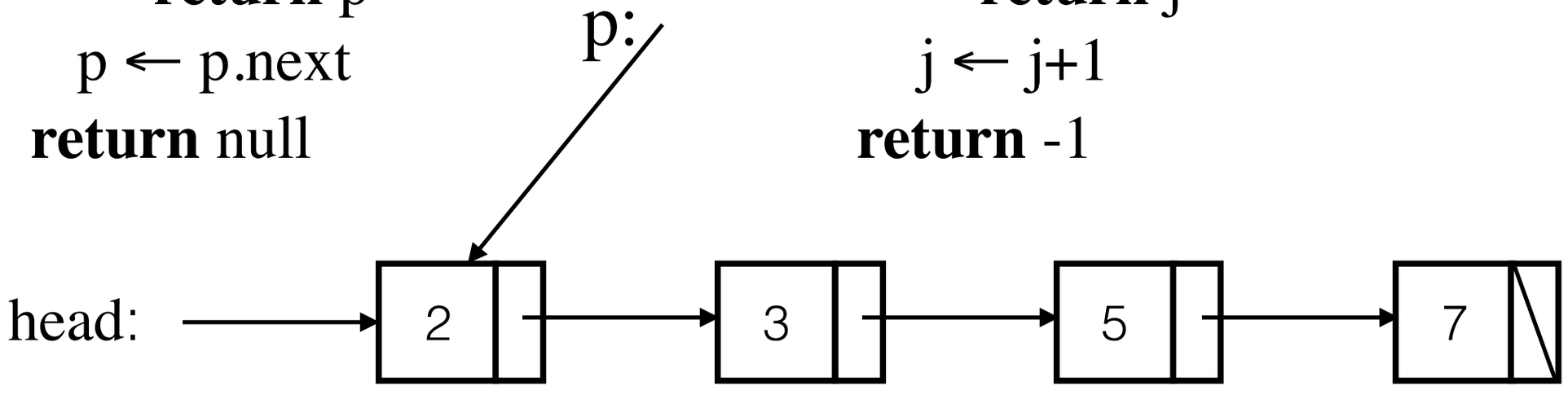
- For example, to locate item **x**.

(note similarity to array version)

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

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

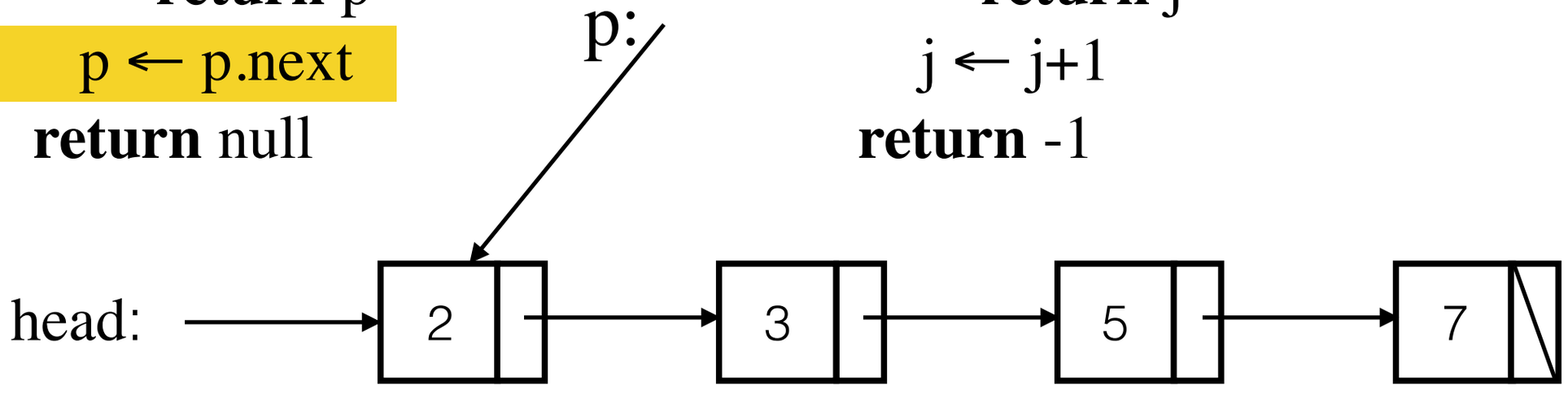
- For example, to locate item **x**.

(note similarity to array version)

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

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

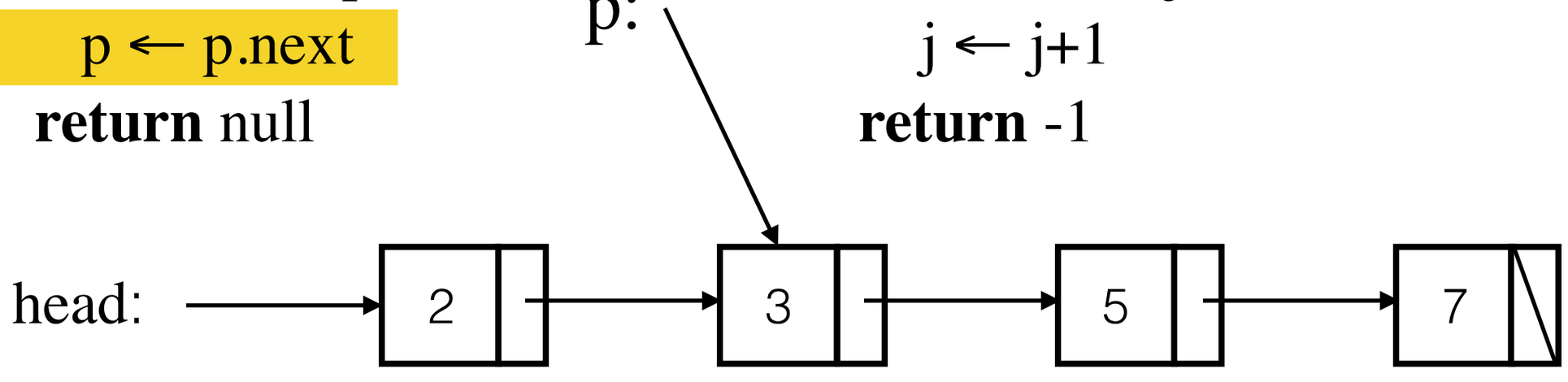
- For example, to locate item **x**.

(note similarity to array version)

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

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

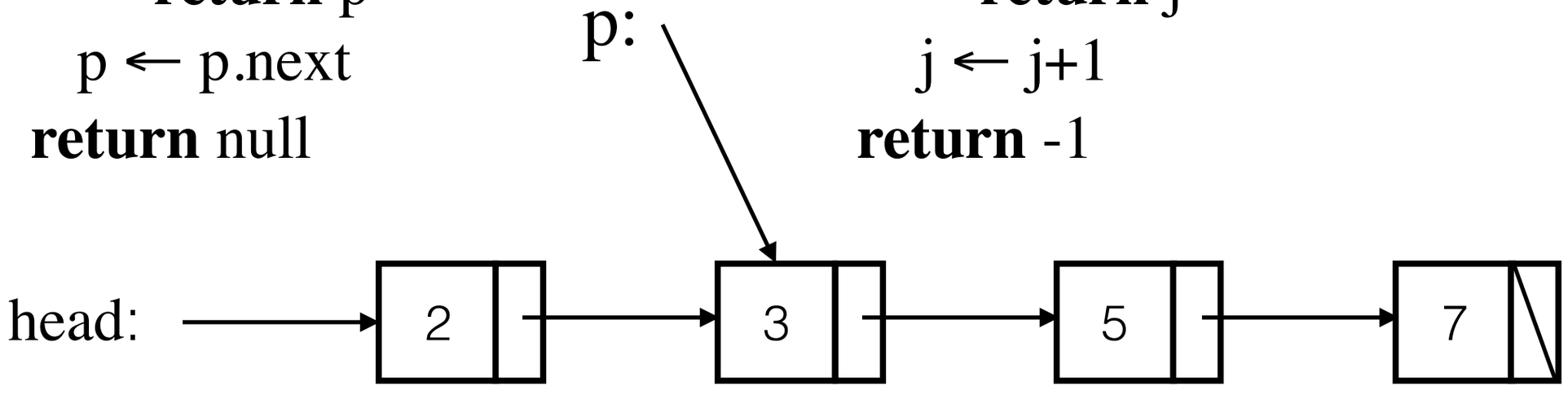
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
  while p ≠ null
    if p.val = x
      return p
    p ← p.next
  return null
```

```
function find(A,x,n)
    i ← 0
  while i < n
    if A[i] = x
      return i
    i ← i+1
  return -1
```

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



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Initial call: find(A,x,0,n-1)

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Initial call: find(A,x,0,n-1)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 0

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 0 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 0 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

A[lo]

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 0 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

A[lo] A[hi]

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 1 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

A[lo] A[hi]

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

A: Y x: 7 lo: 1 hi: 6

Assignment Project Exam Help

https://powcoder.com

A[lo] A[hi]

Add WeChat powcoder

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

A: Y x: 7 lo: 1 hi: 6

Assignment Project Exam Help

https://powcoder.com

A[lo] A[hi]

Add WeChat powcoder

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

A: Y x: 7 lo: 2 hi: 6

Assignment Project Exam Help

https://powcoder.com

A[lo] A[hi]

Add WeChat powcoder

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

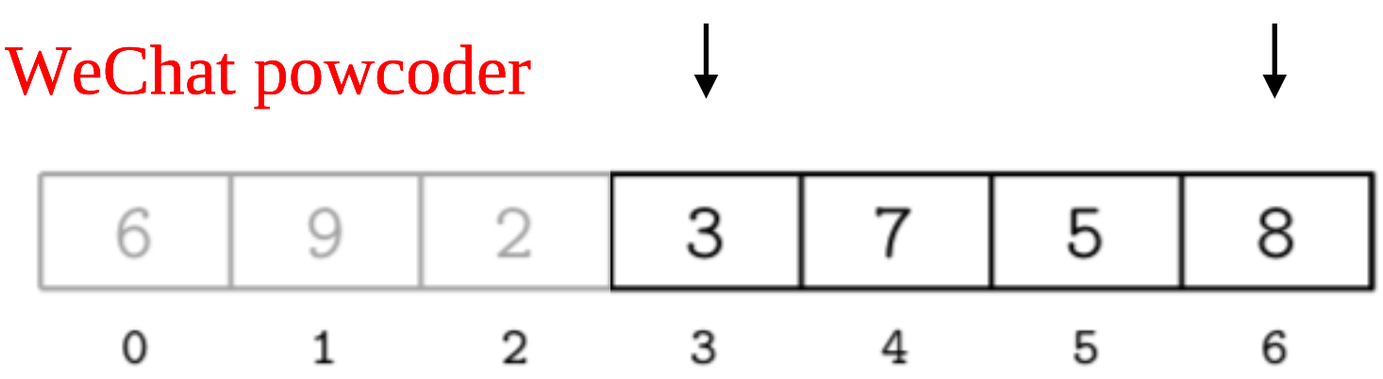
- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 3 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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



Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

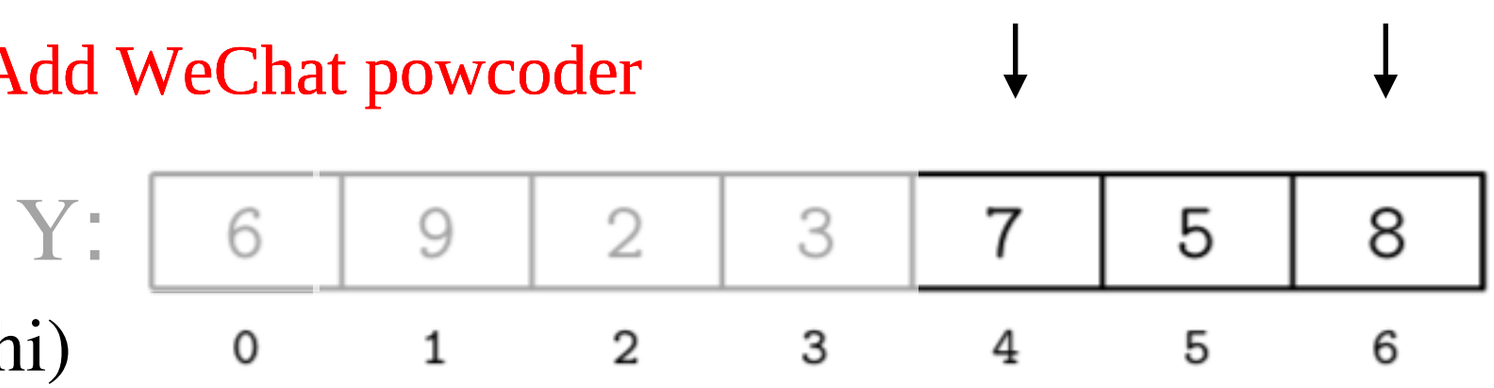
A: Y x: 7 lo: 4 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

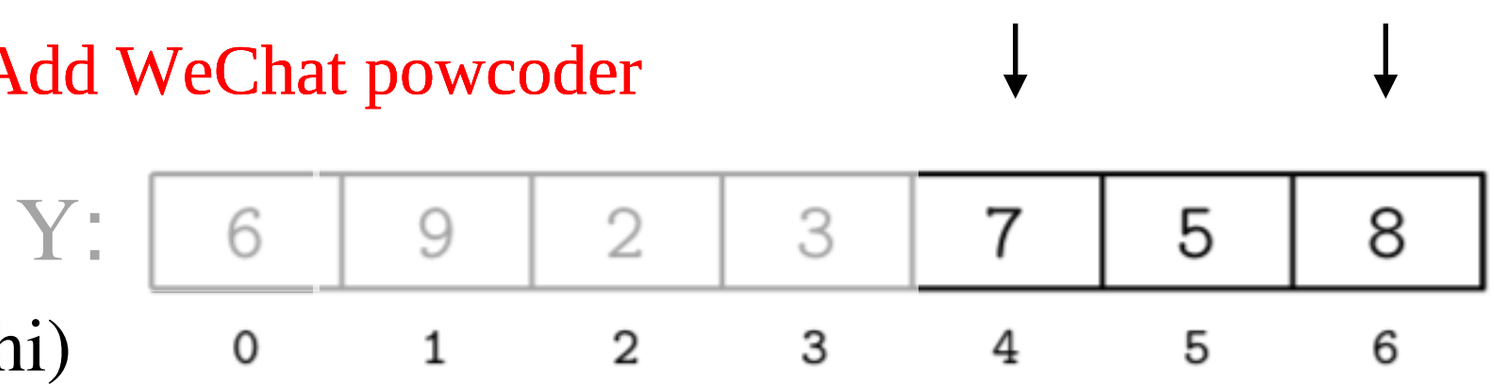
A: Y x: 7 lo: 4 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

(returns 4)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

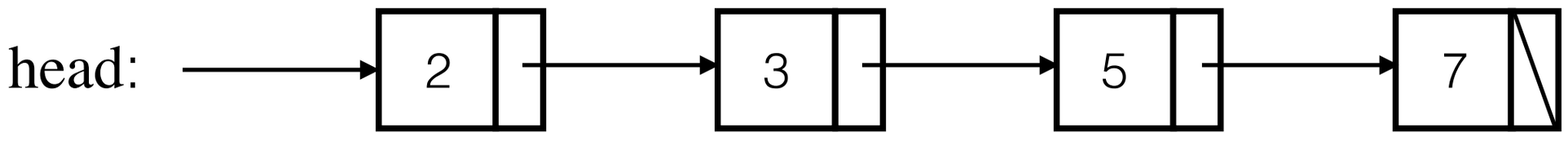
- Solve the problem for a sub-instance and use the solution to solve the full instance

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

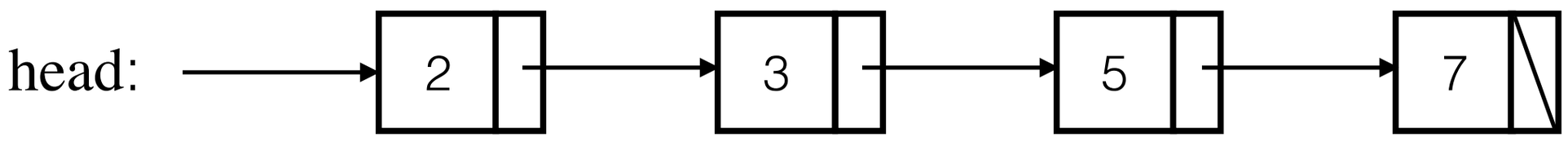
```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

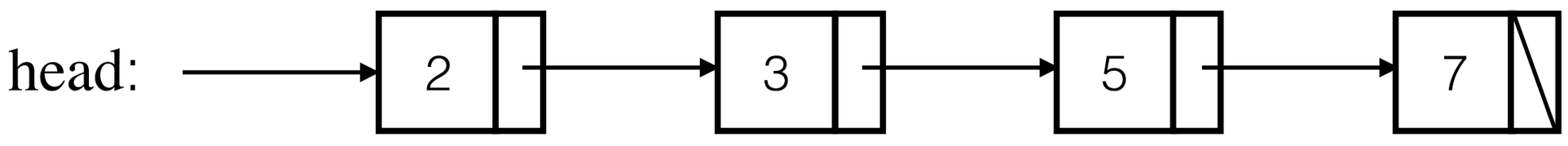
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

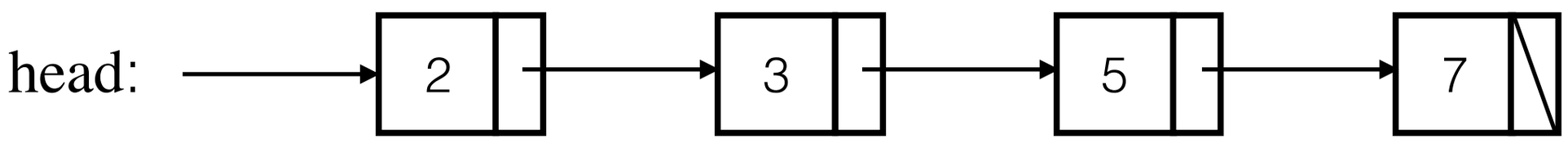
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

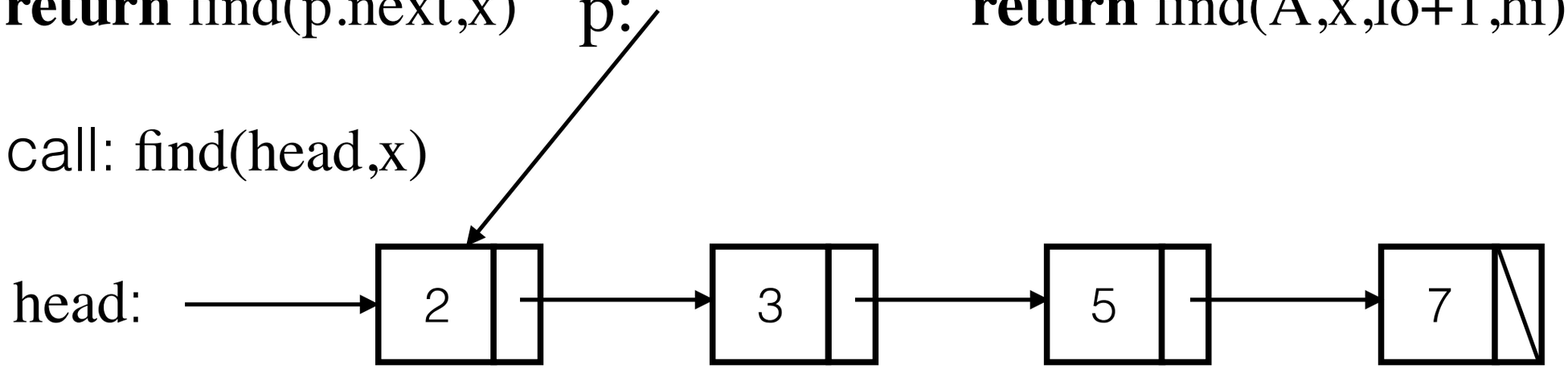
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

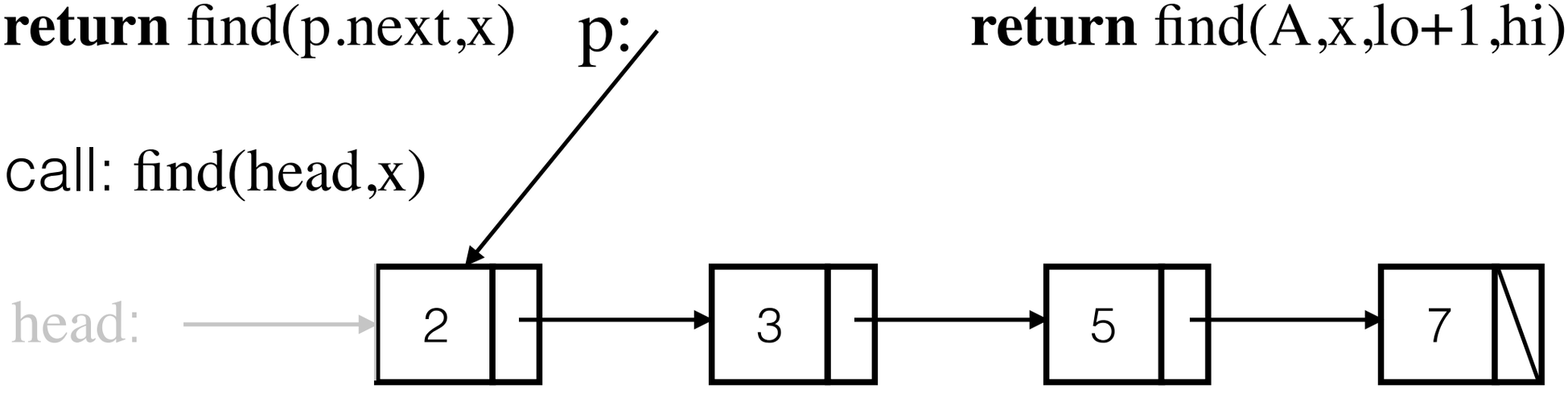
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

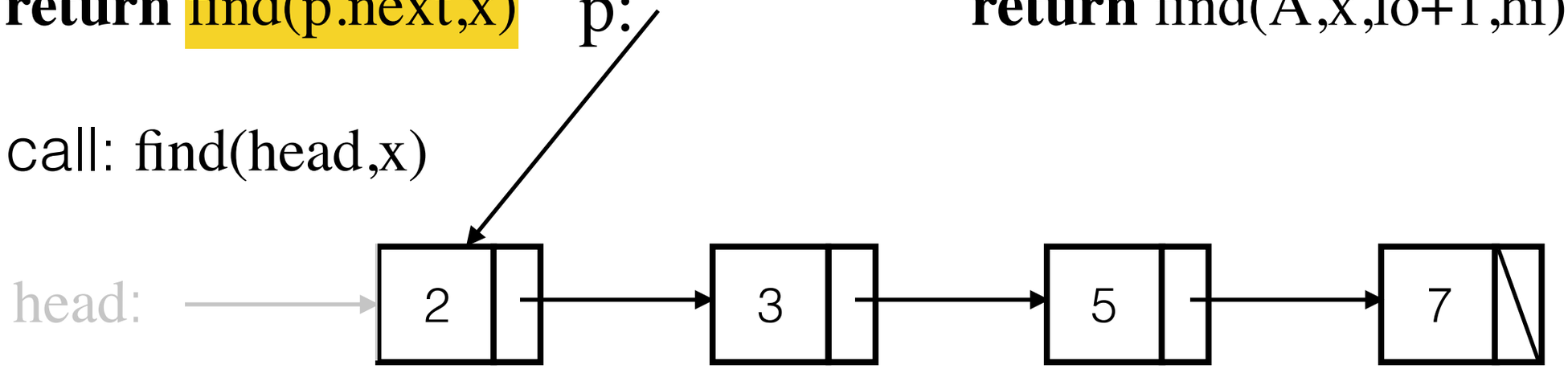
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

THE UNIVERSITY OF MELBOURNE

- Solve the problem for a sub-instance and use the solution to solve the full instance

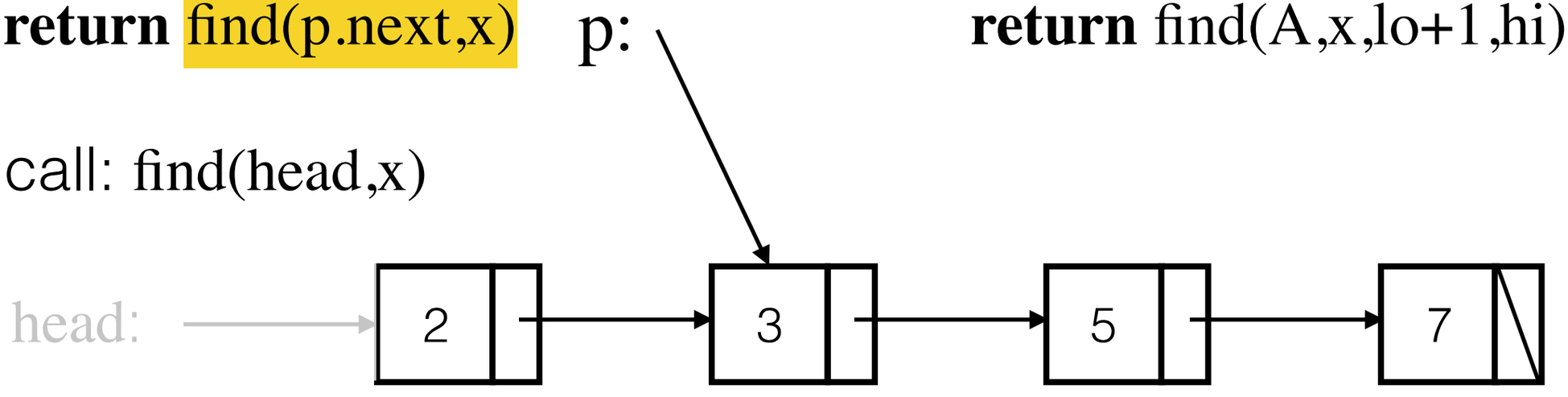
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

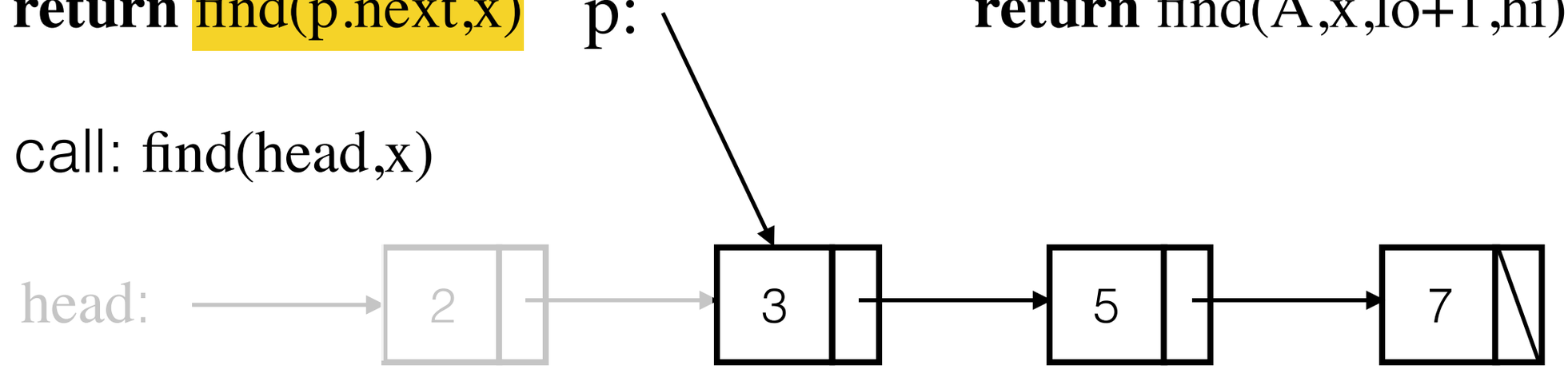
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

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

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

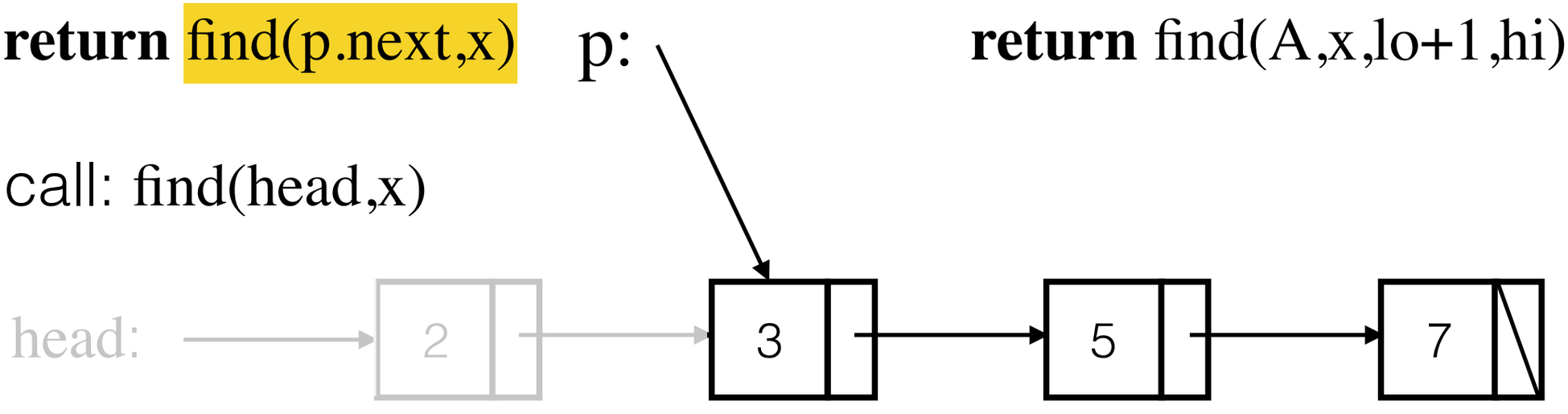
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

We will discuss recursion properly in Week 3
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Abstract DataTypes

THE UNIVERSITY OF MELBOURNE

- A collection of data items, and a family of operations that operate on that data
- Think of an ADT as a set of contracts, an **interface**
- We must still **implement** these promises, but it is an advantage to separate the implementation of the ADT from the "concept" (i.e. the interface it provides)

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

- Good programming practice is to support this separation
  - Nothing outside of the definition of the ADT should refer to anything inside, except through function calls and basic operations

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

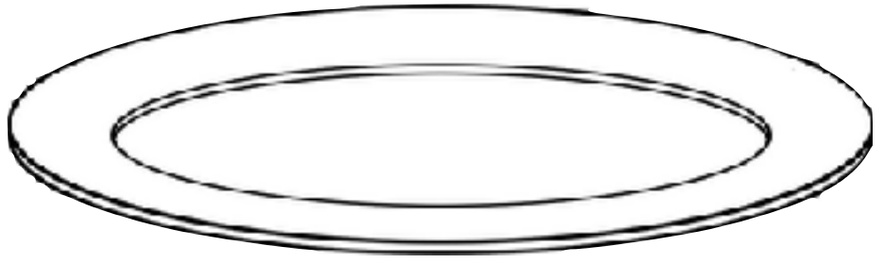
THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



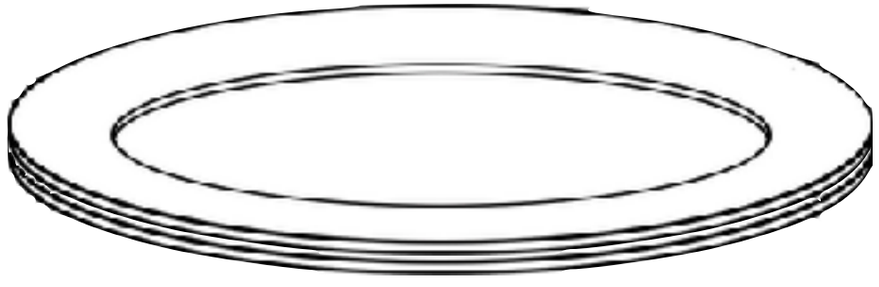
 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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

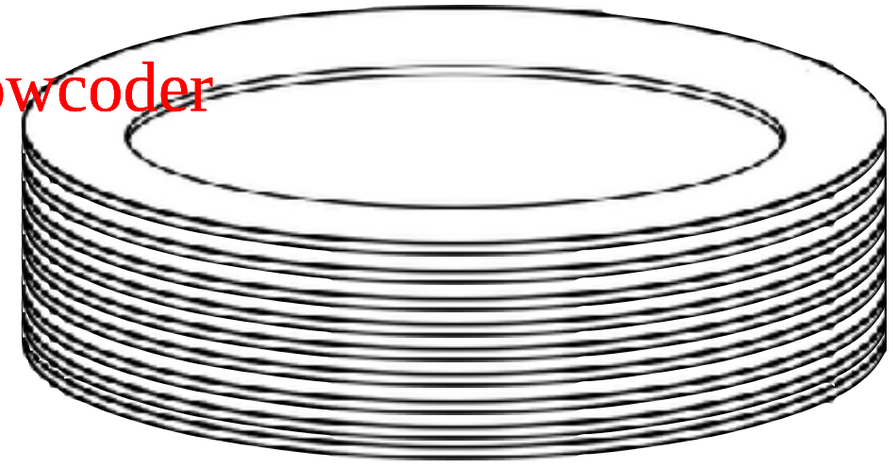


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



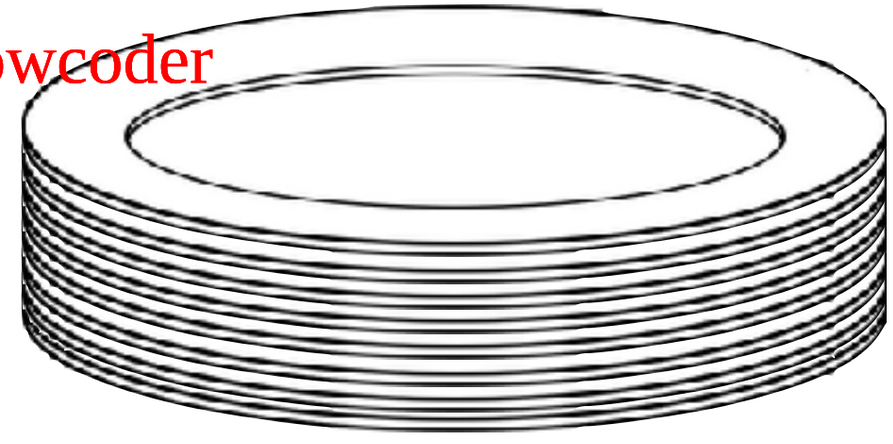
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



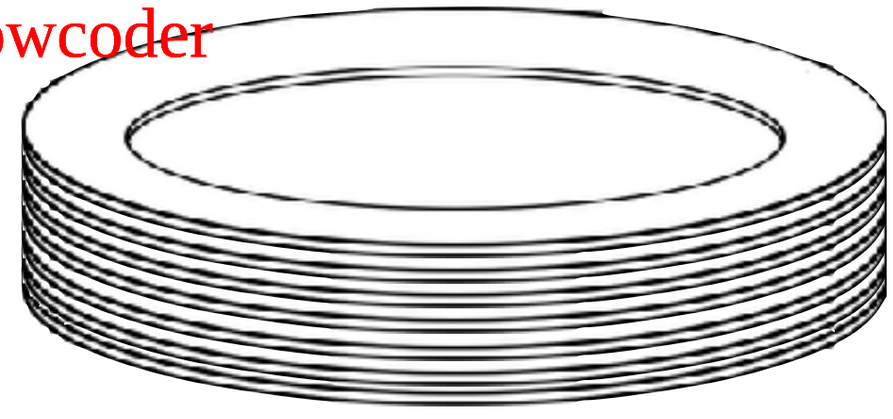
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



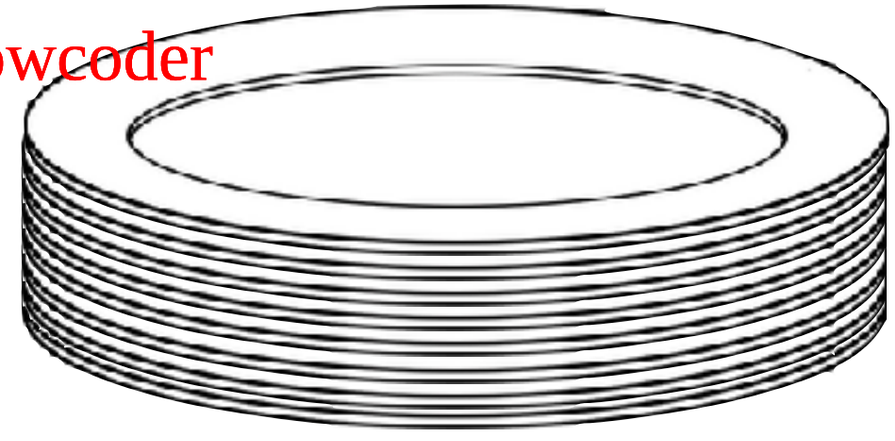
 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



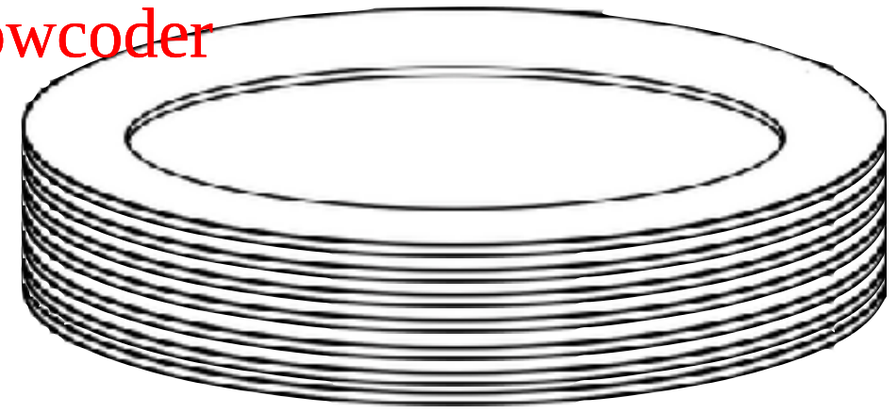
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



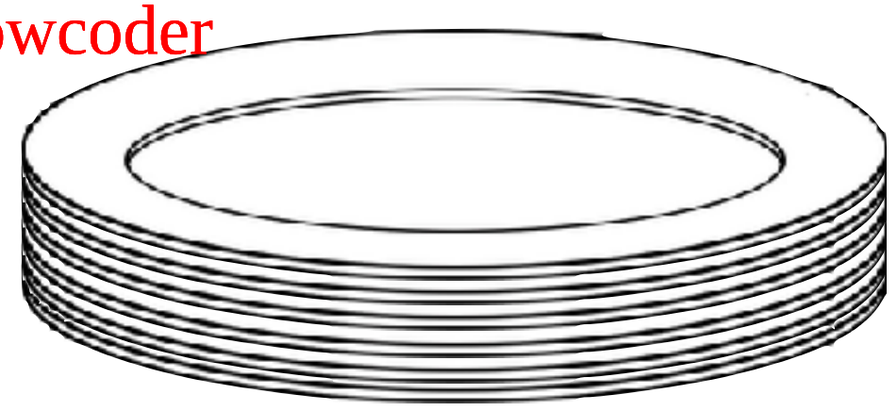
 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

THE UNIVERSITY OF MELBOURNE

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

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



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

THE UNIVERSITY OF MELBOURNE

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

top: 5

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

top: 5

Push(5)

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

top: 5

Push(5)

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

Assignment Project Exam Help

top: 6

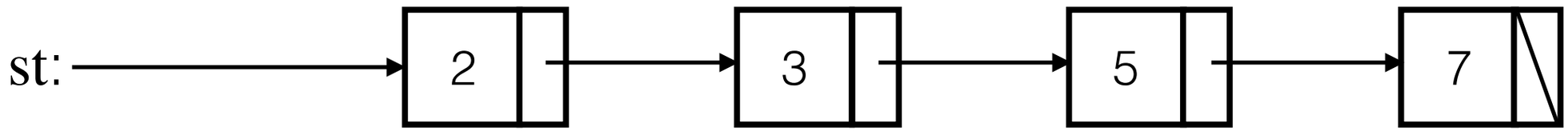
https://powcoder.com

Push(5)

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List

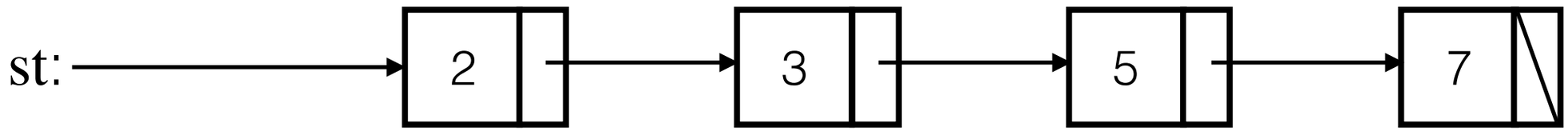


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



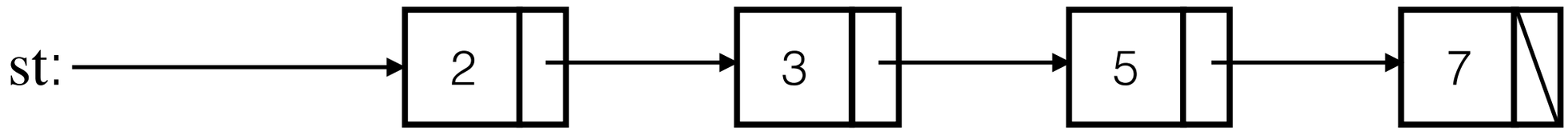
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



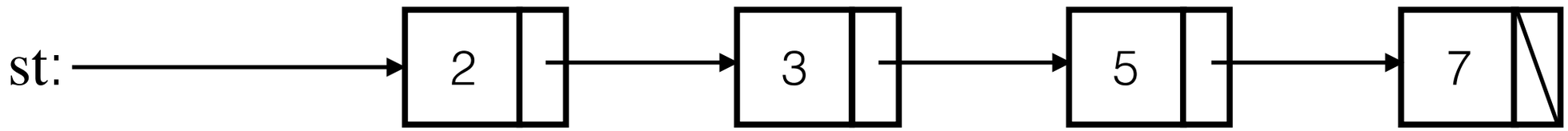
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

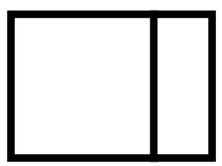
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





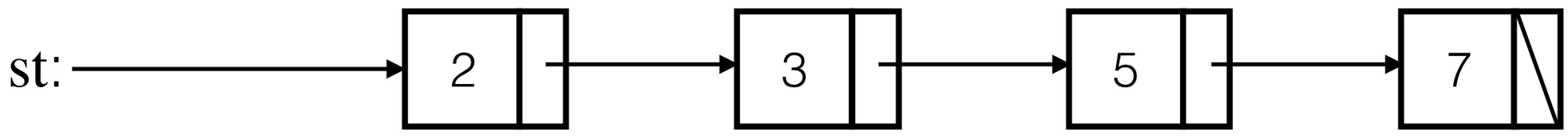
Push(5)

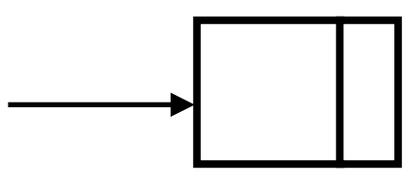
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





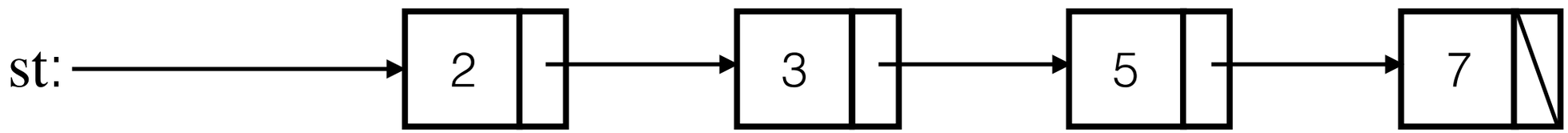
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

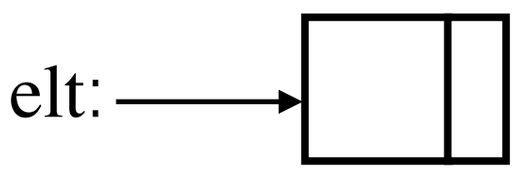
Push(5)

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





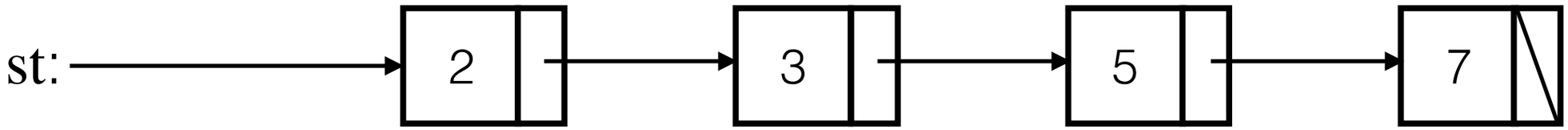
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

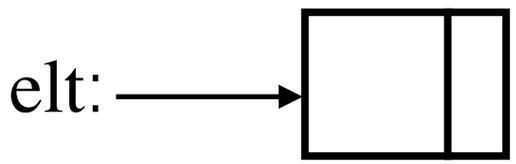
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





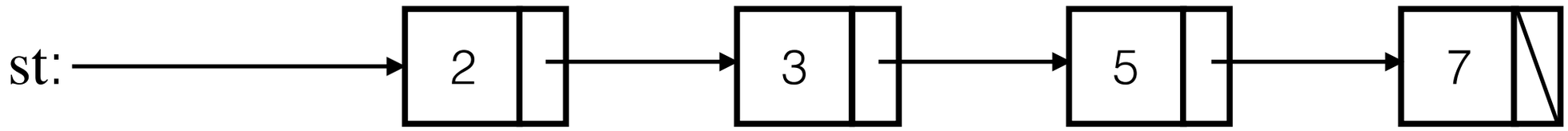
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

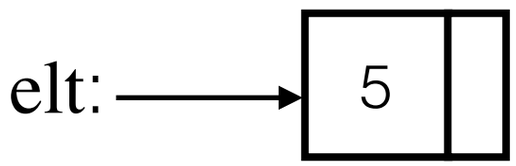
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





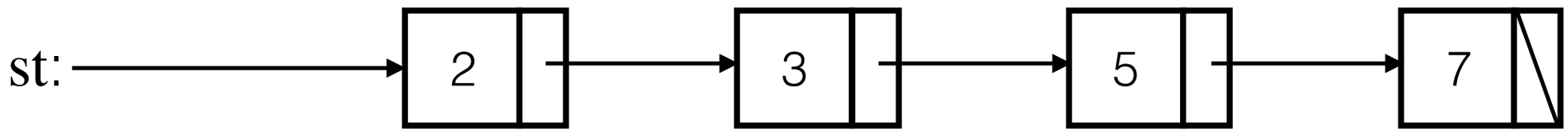
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

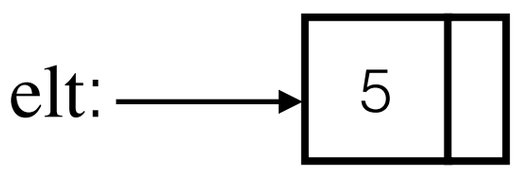
```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





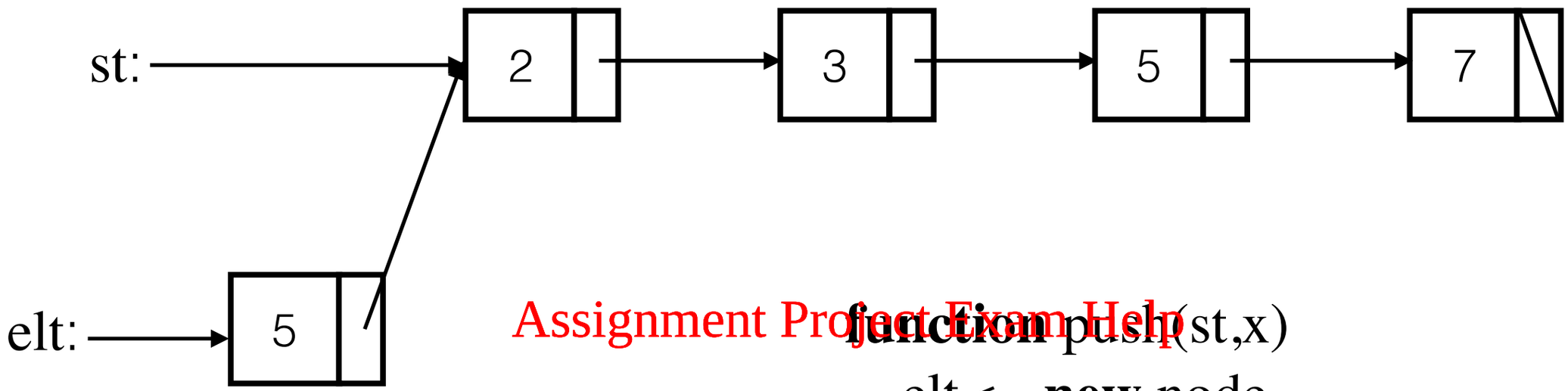
Push(5)

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

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```


Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



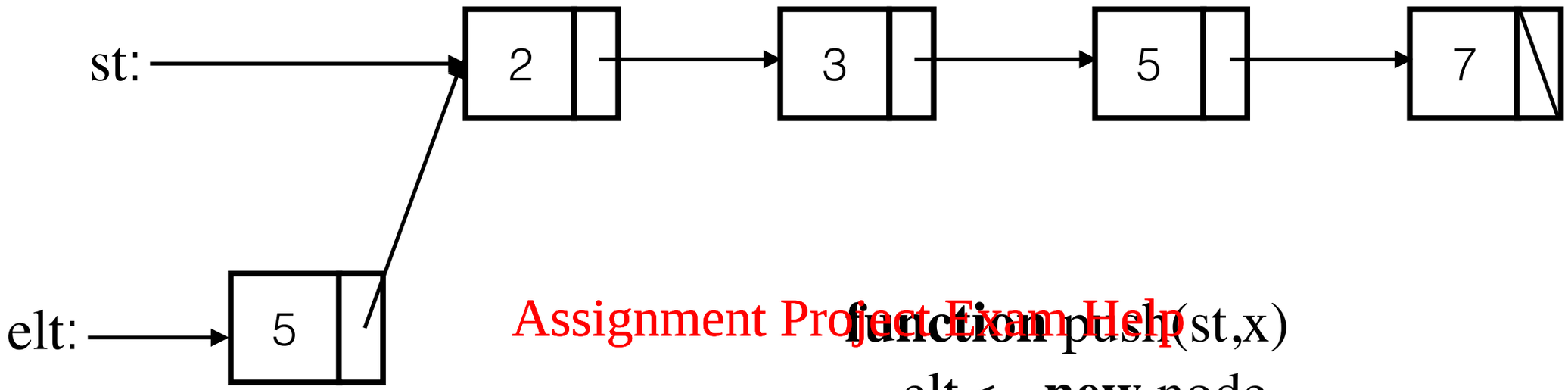
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



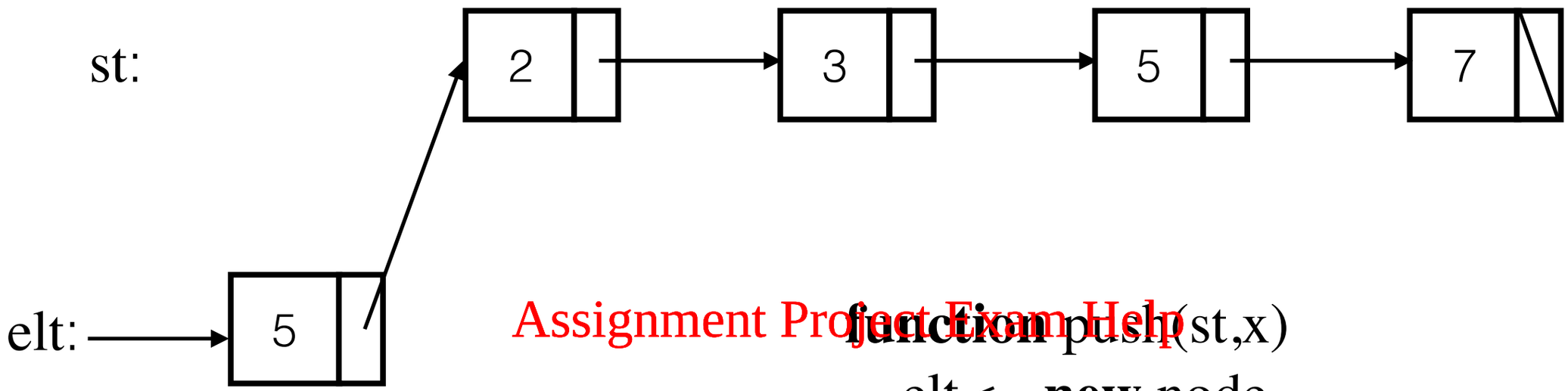
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



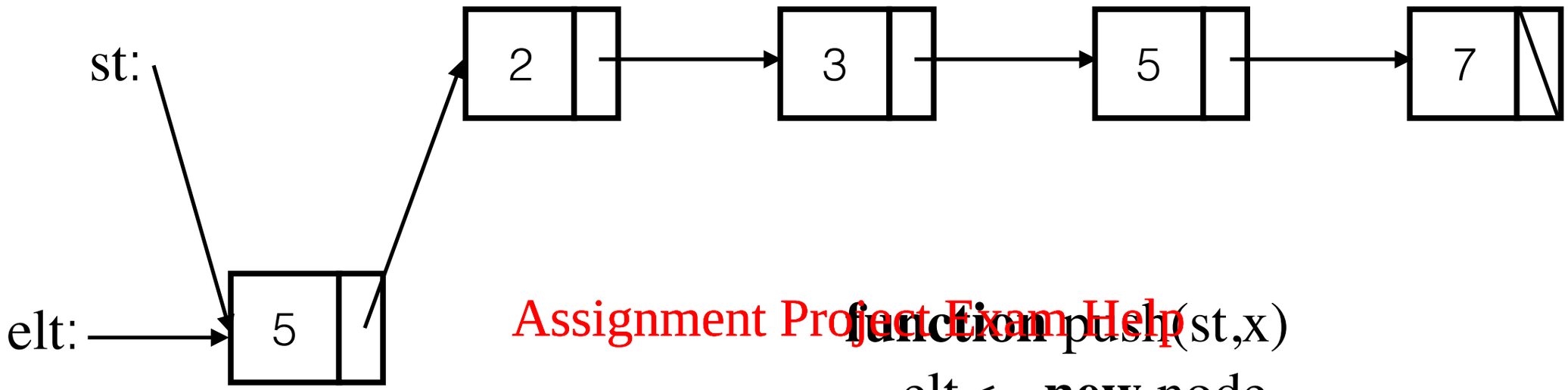
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



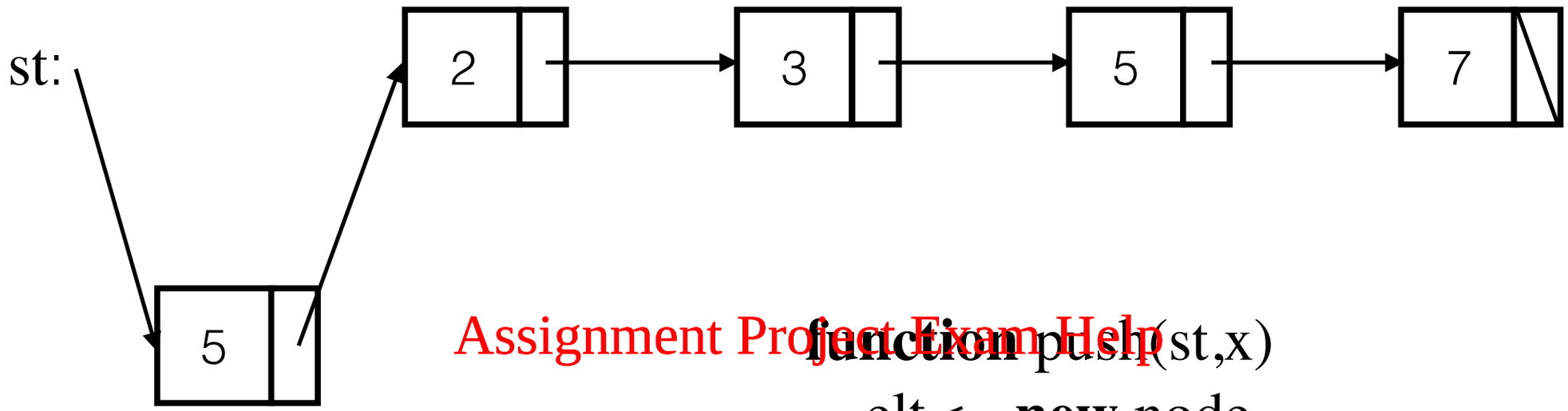
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



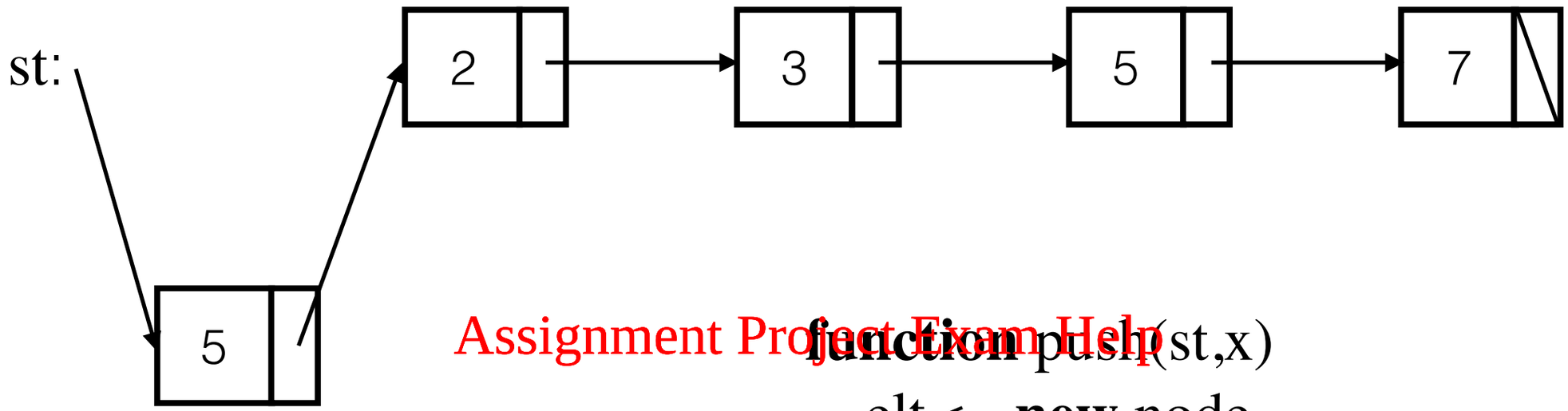
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



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

```
function push(st,x)
    elt ← new node
    elt.val ← x
    elt.next ← st
    st ← elt
    return st
```

Push(5)

See
https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
for more visualisations

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Pseudo Code

THE UNIVERSITY OF MELBOURNE

- On the previous slide, we assumed that a "node" has two attributes: a "val" which is its value, and a "next" which points to the rest of the list.

Assignment Project Exam Help

https://powcoder.com

- There is no standard for pseudo-code. Use the examples in Levitin as a guide. Cormen et al. pages 20–22 (in Reading Resources) has a list of standard conventions used with pseudo-code which are good to follow, except we use ← as the assignment operator.

Add WeChat powcoder

 Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

THE UNIVERSITY OF MELBOURNE

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

THE UNIVERSITY OF MELBOURNE

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

THE UNIVERSITY OF MELBOURNE

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

THE UNIVERSITY OF MELBOURNE

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

THE UNIVERSITY OF MELBOURNE

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Other Data Structures

- We will meet many other (abstract) data structures, e.g.
  - The priority queue
  - Various types of “tree”
  - Various types of “graph”
- If you check out algorithm animation tools or advanced algorithm books, you will meet exotic data structures such as splay trees and skip lists.

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

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Next Week

Assignment Project Exam Help

https://powcoder.com

- Algorithm analysis—how to reason about an algorithm's resource consumption.

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License