Data Structures

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

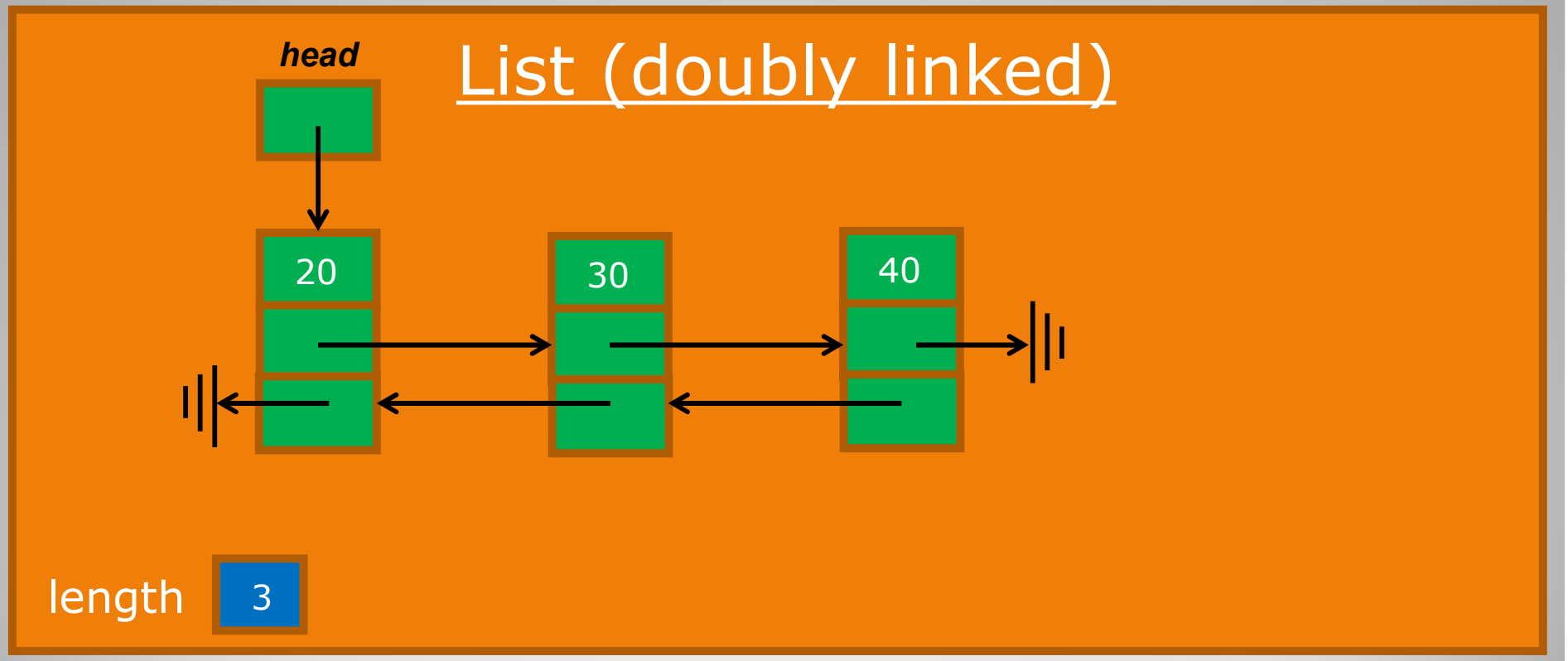
- Circular doubly-linked lists (ordered)

Today's Lecture

- In a normal doubly linked list, the last node's next pointer has the value null.
- The first node's previous pointer has the value null.
- For example…

Normal Doubly Linked List

- Last node's next points to null.
- First node's previous points to null.

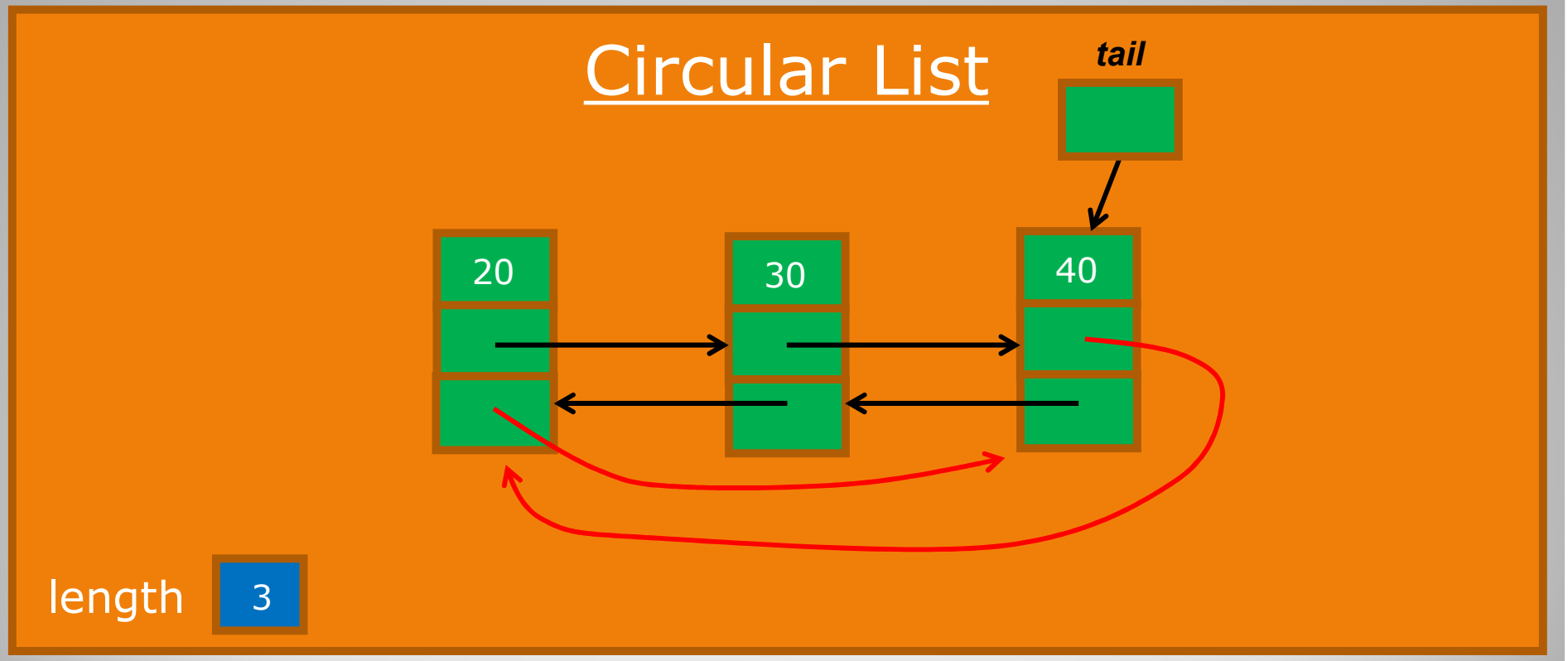


Doubly-linked List

Circular List

- A circular list has the last element point to the first element instead of null.
- We would like to be able to easily get both the head and tail nodes.
- We will only keep a tail pointer instead of a head pointer.
- For example…

- We are only using a tail pointer in this example so we can easily get to the head or tail node.
- Last node's next points to first.
- First node's previous points to last.



Circular Doubly-linked List

- How do a get a pointer to the start of the list (the head)?

Circular List – Start pointer

Circular List – Start pointer

- How do a get a pointer to the start of the list?
- **ANSWER**

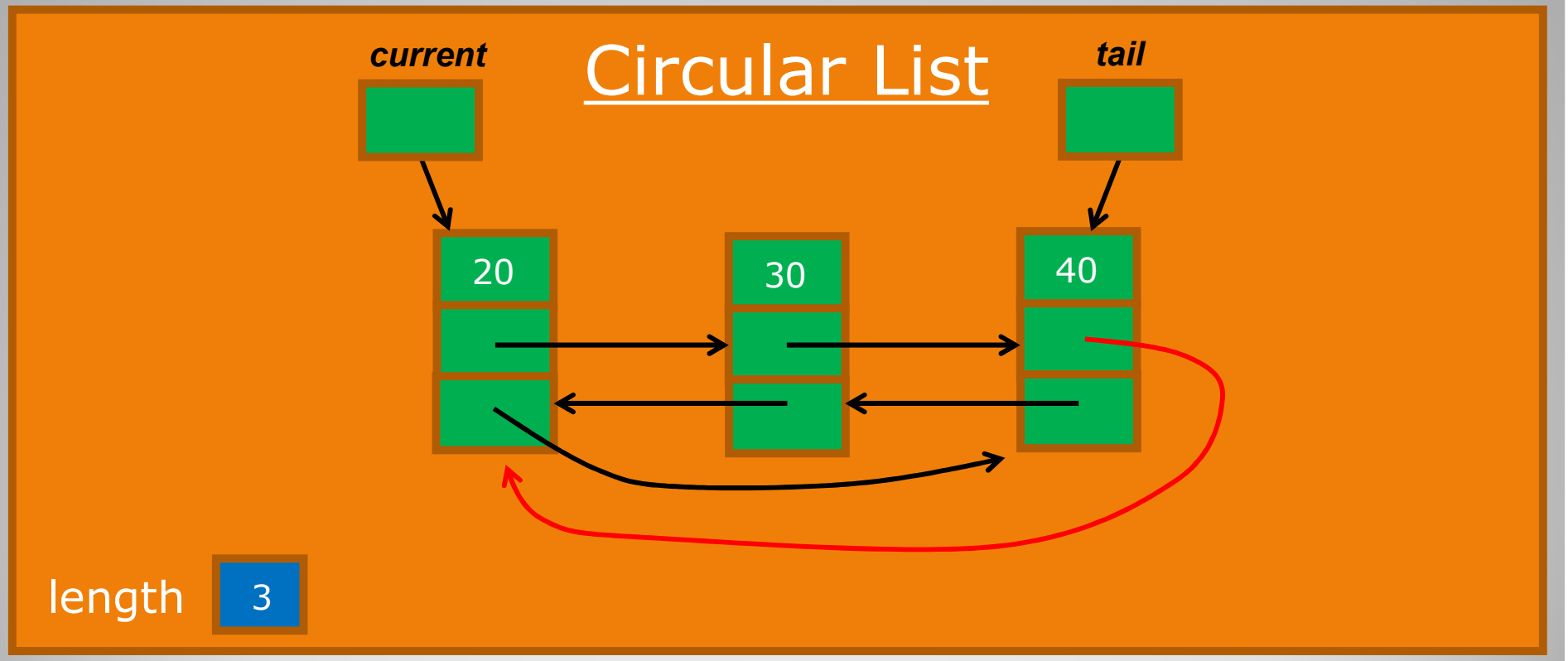
The tail's next pointer points to the start of the list.

```
// current will point to the start of the list (head)
Node current = tail.next;
```

Here is a picture…

Head Node

- The head node is tail's next.
- Current now points the starting node (the head).



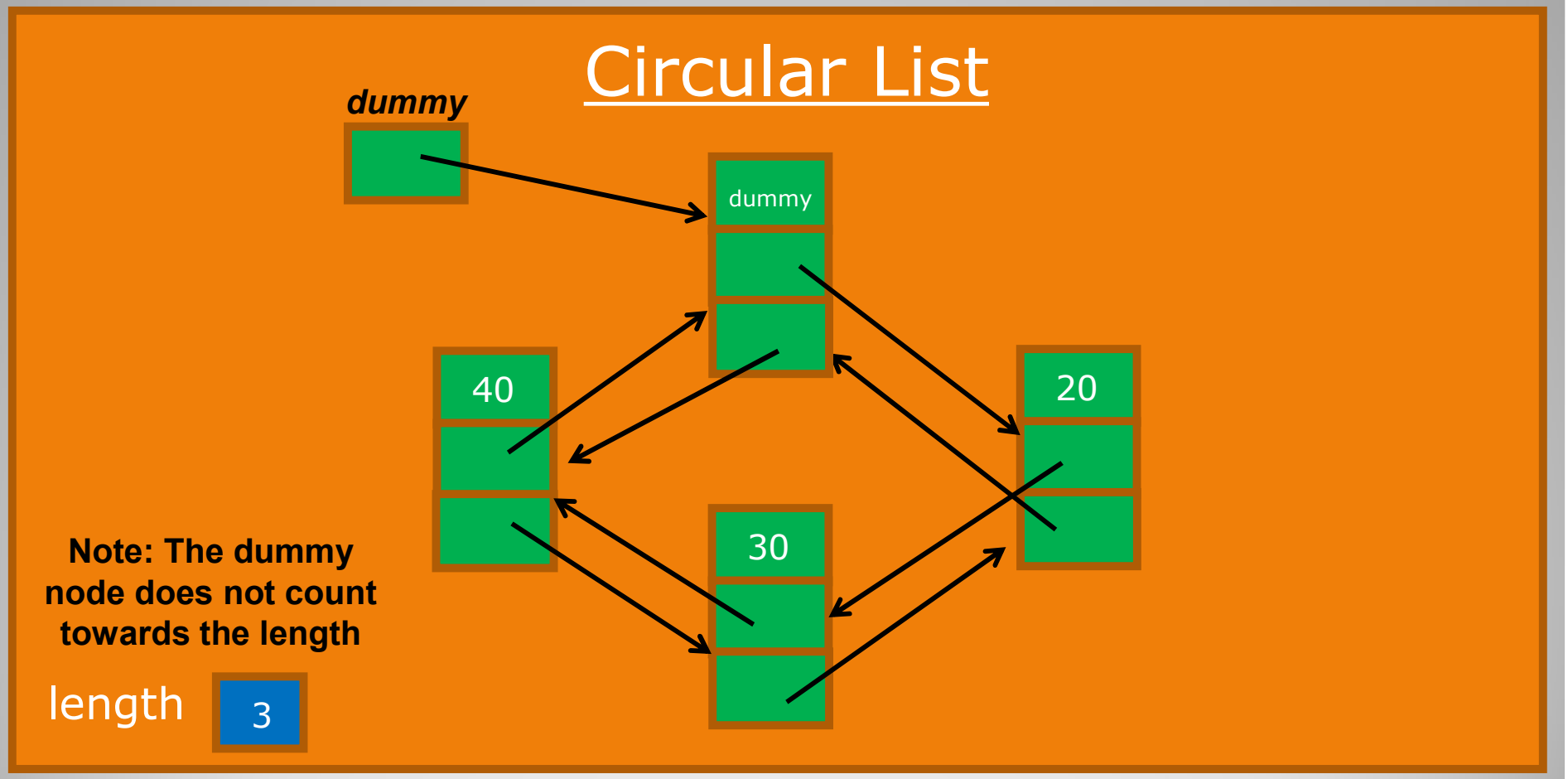
Circular Doubly-linked List

DUMMY NODE - BETTER SOLUTION THAN A HEAD OR TAIL

- Instead of keeping a head or tail reference we can keep a reference to a dummy node.
- A dummy node makes it easy to code special cases for the list operations.
- A dummy node is a node in the list, but its data does not mean anything (its data is essentially garbage).
- For example…

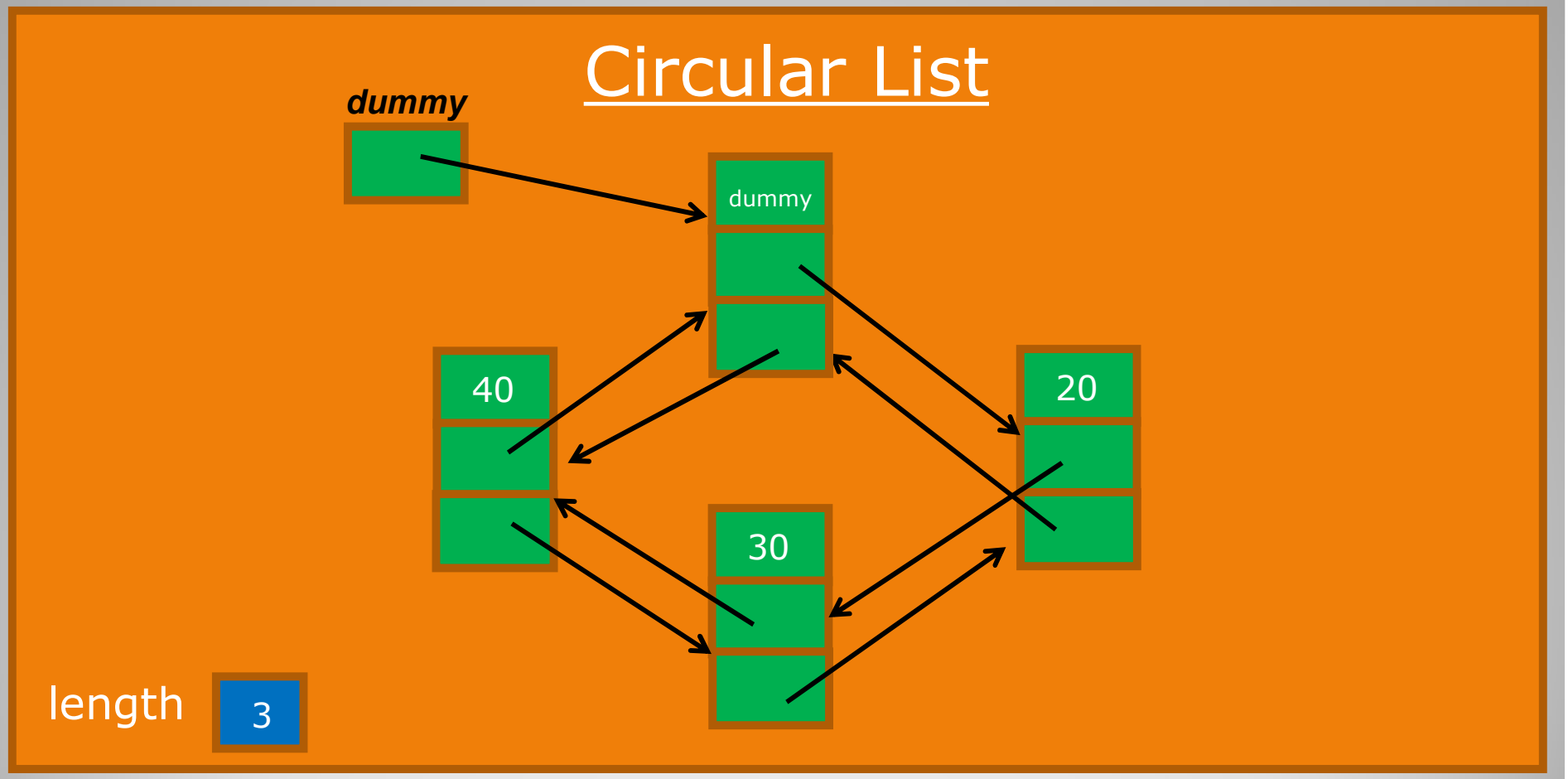
Circular Doubly-linked List w/Dummy

- Enter the list through the dummy node.
- You can get to the head or tail through the dummy node.



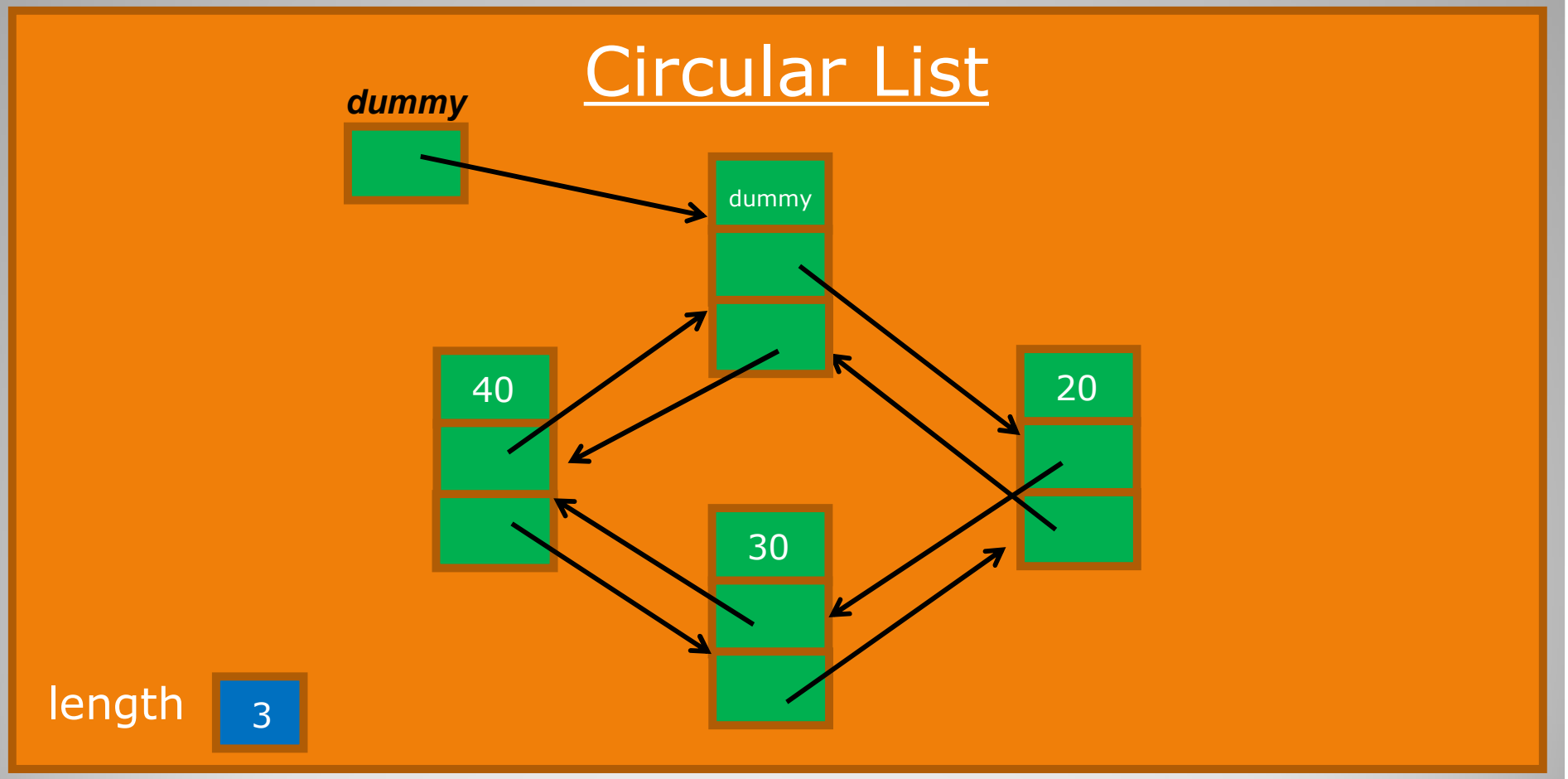
Circular Doubly-linked List w/Dummy

- What is the head?
- What is the tail?



Circular Doubly-linked List w/Dummy

- What is the head? head = dummy.next;
- What is the tail? tail = dummy.previous;



Circular Doubly-linked List w/Dummy

- Now we will cover insertions into a circular doubly-linked list with a dummy…

Circular Doubly-linked List w/Dummy - insertions

- We will allow insertions at both the head and the tail.
- The dummy node makes it easy to do insertions in either place.
- The dummy node basically removes special cases.

Circular Doubly-linked List w/Dummy - insertions

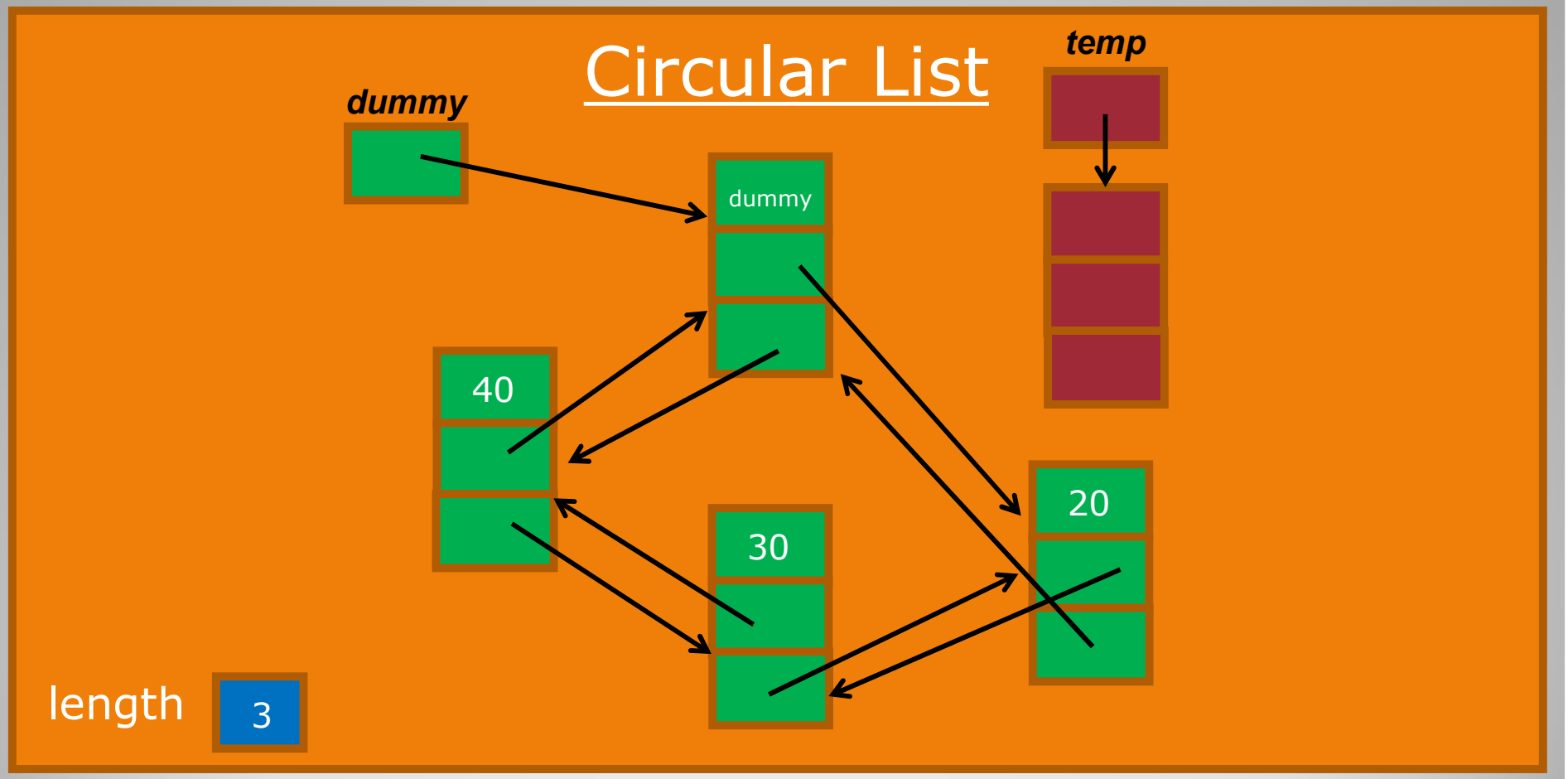
insertHead Pseudocode

1. Create a new Node instance named temp (dynamically allocate).
2. Set the fields on the new Node.
   A. Set the data.
   B. Set the next pointer to dummy's next.
   C. Set the prev pointer to dummy.
3. Set temp.previous.next to temp.
4. Set temp.next.previous to temp.
5. Increment the length of the list.

Circular Doubly-linked List w/Dummy - insertHead

cl.insertItem(10);

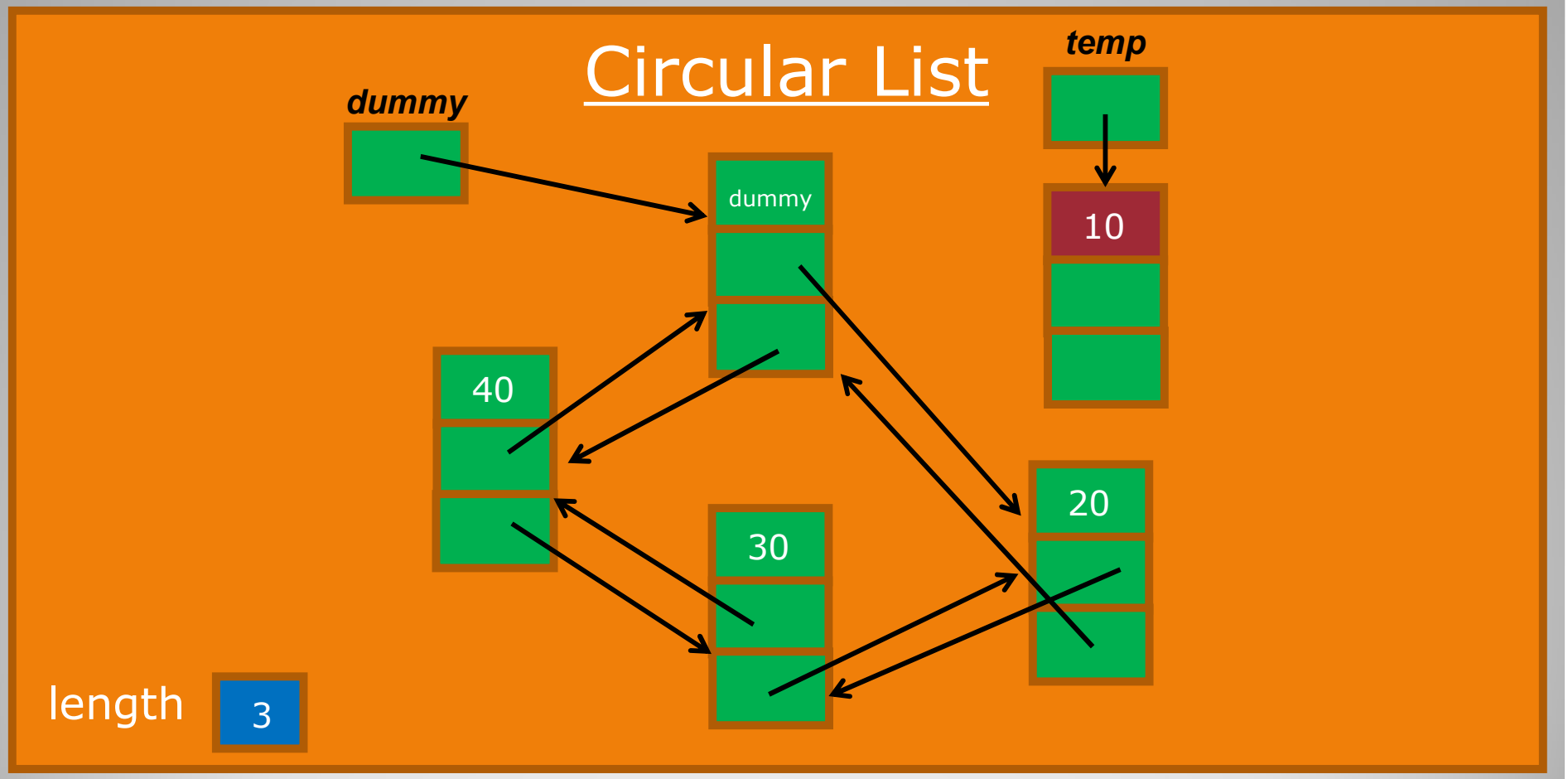
1. Create a new Node instance named temp (dynamically allocate).



Circular -insertHead

cl.insertItem(10);

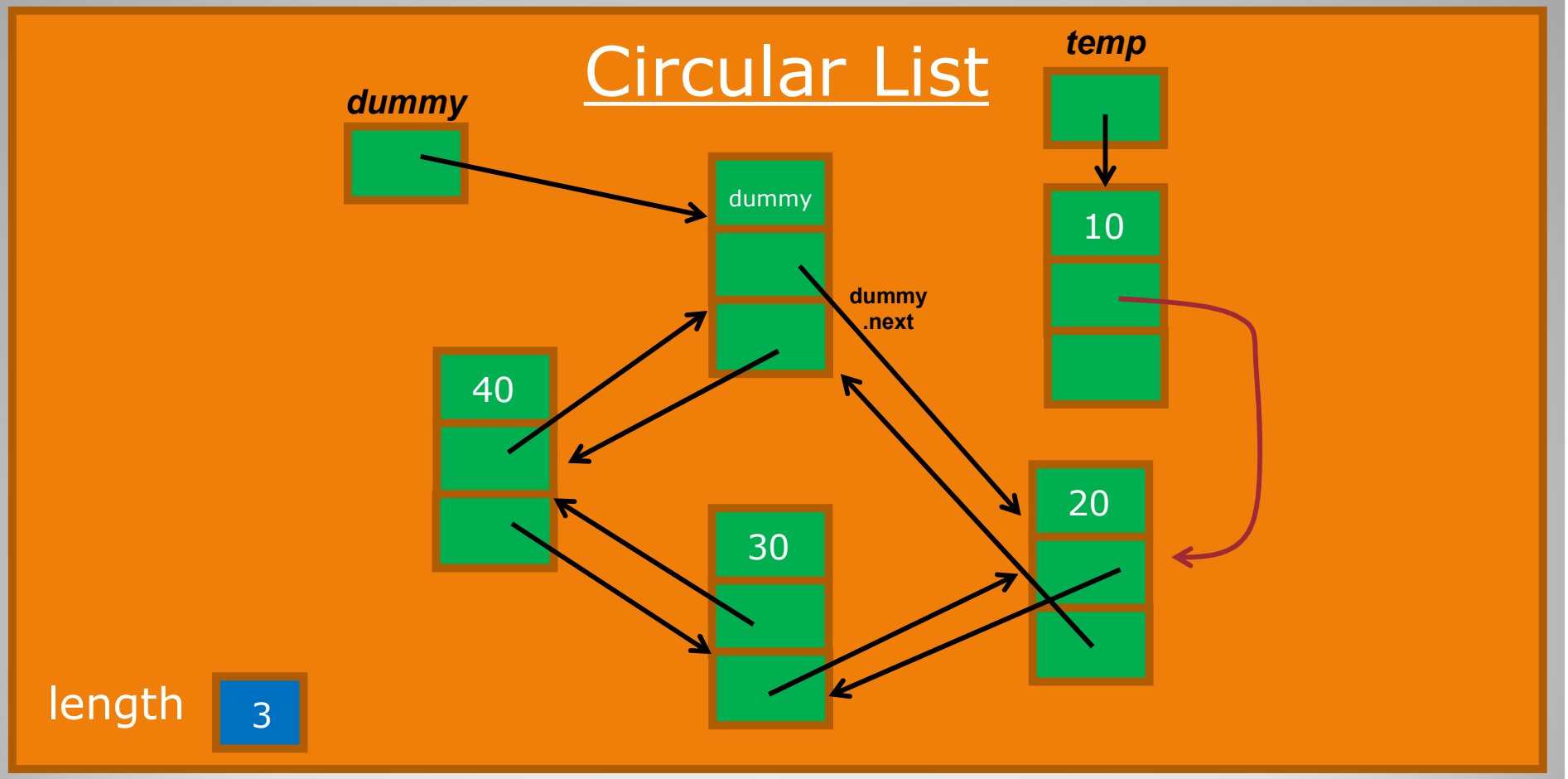
2. New node fields – A. Set the data



Circular -insertHead

cl.insertItem(10);

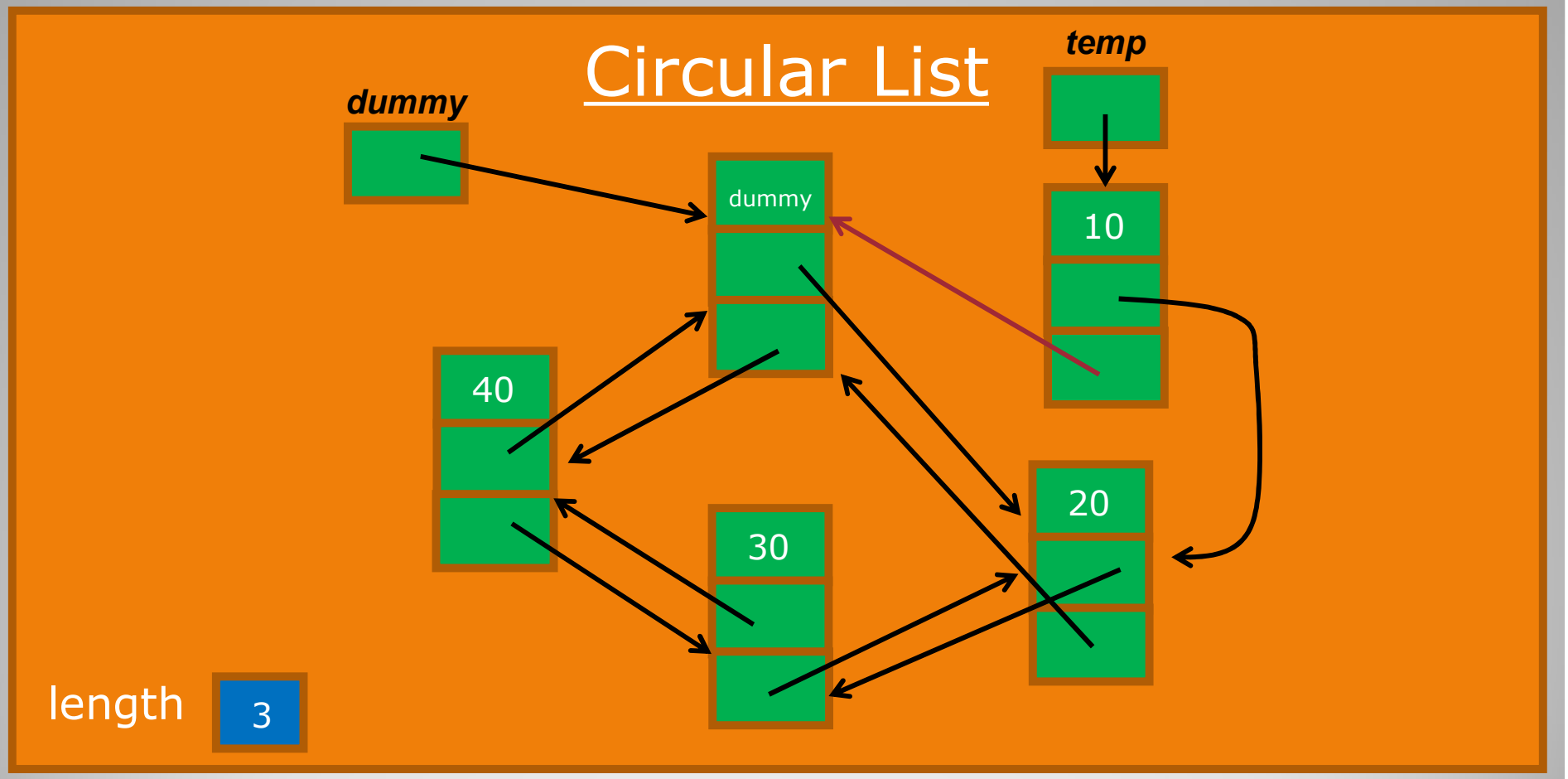
2. New node fields – B. Set the next pointer to dummy's next.



Circular -insertHead

cl.insertItem(10);

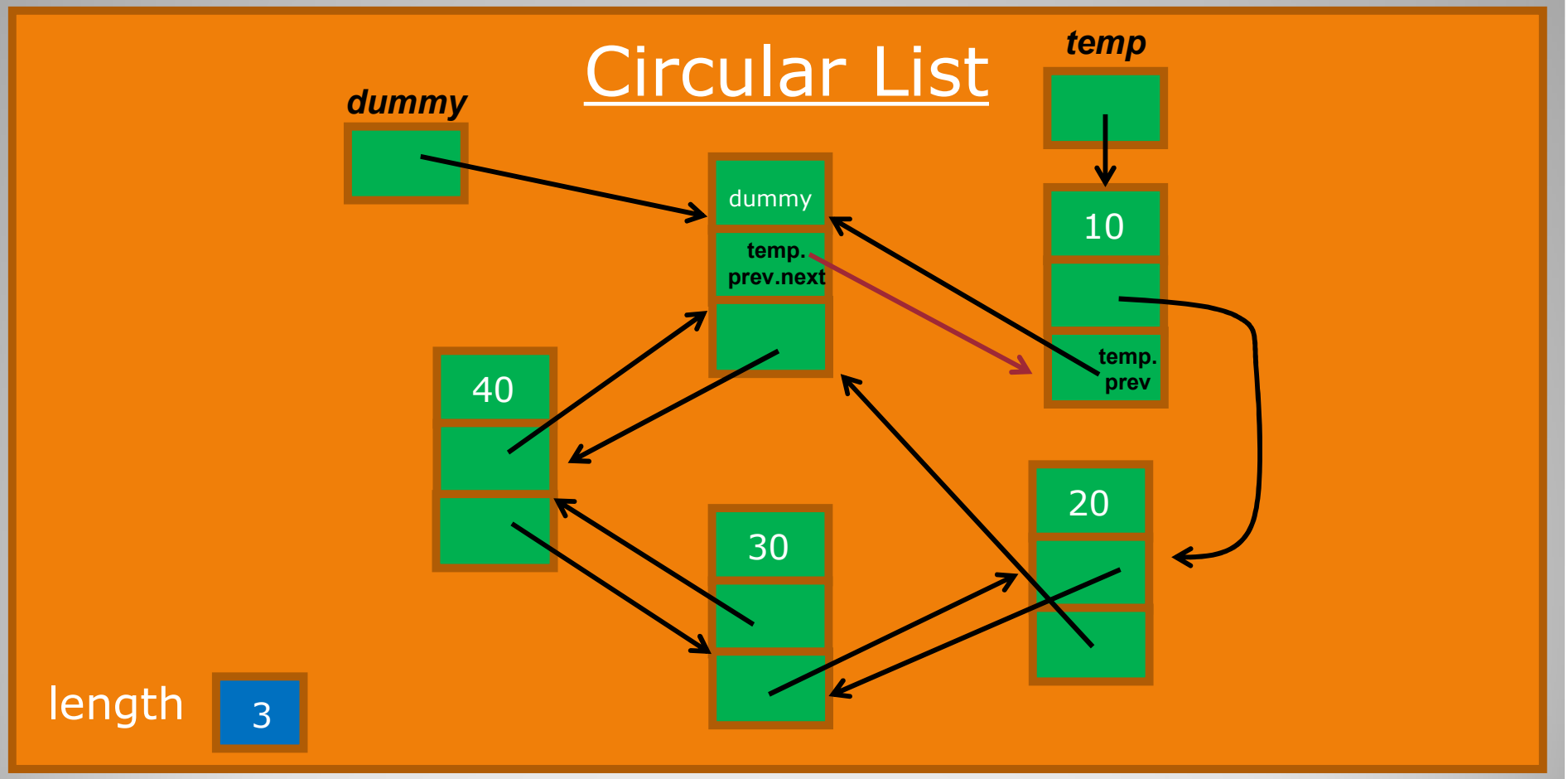
2. New node fields – C. Set the prev pointer to dummy



Circular -insertHead

cl.insertItem(10);

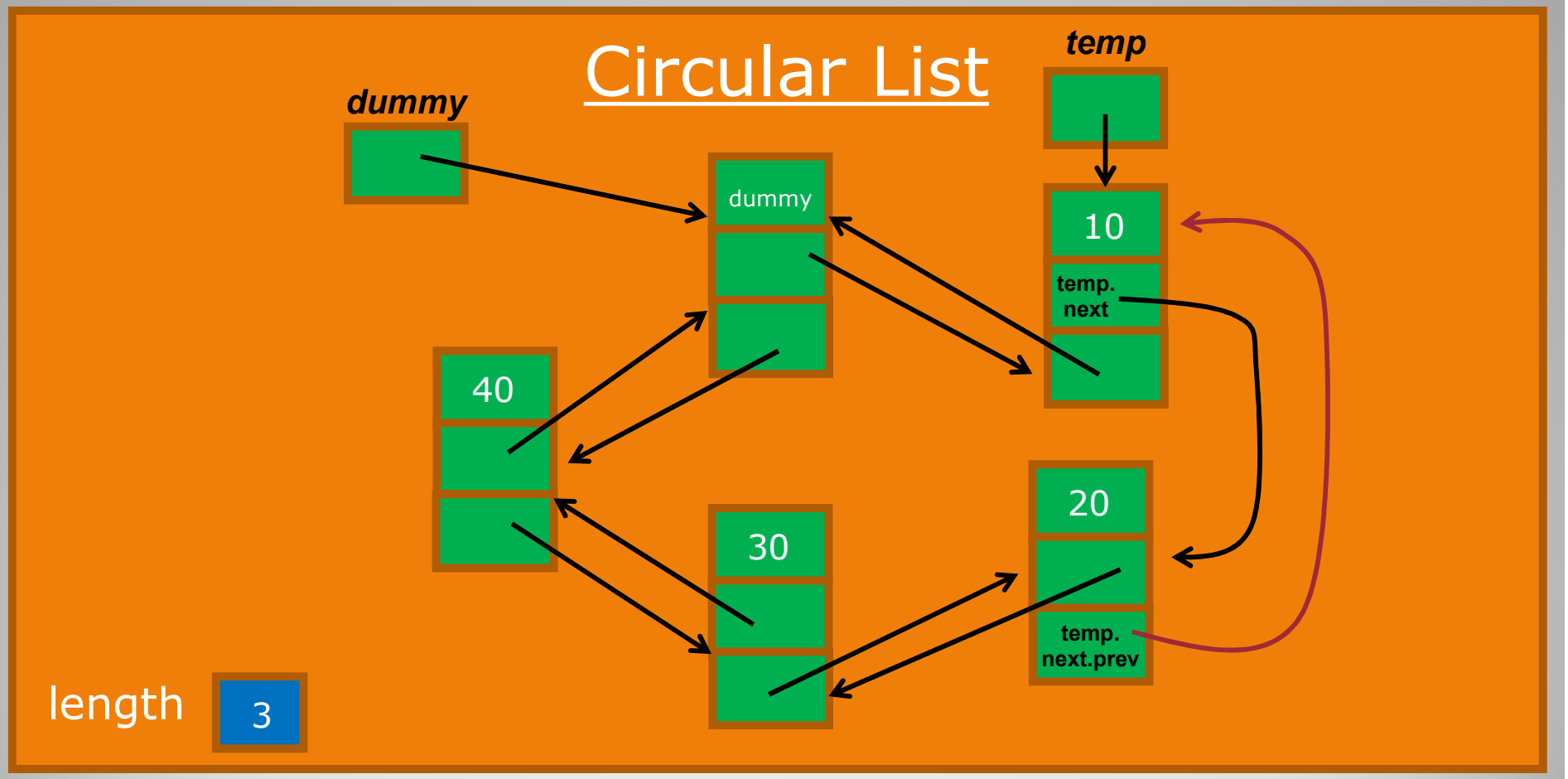
3. Set temp.previous.next to temp.



Circular -insertHead

cl.insertItem(10);

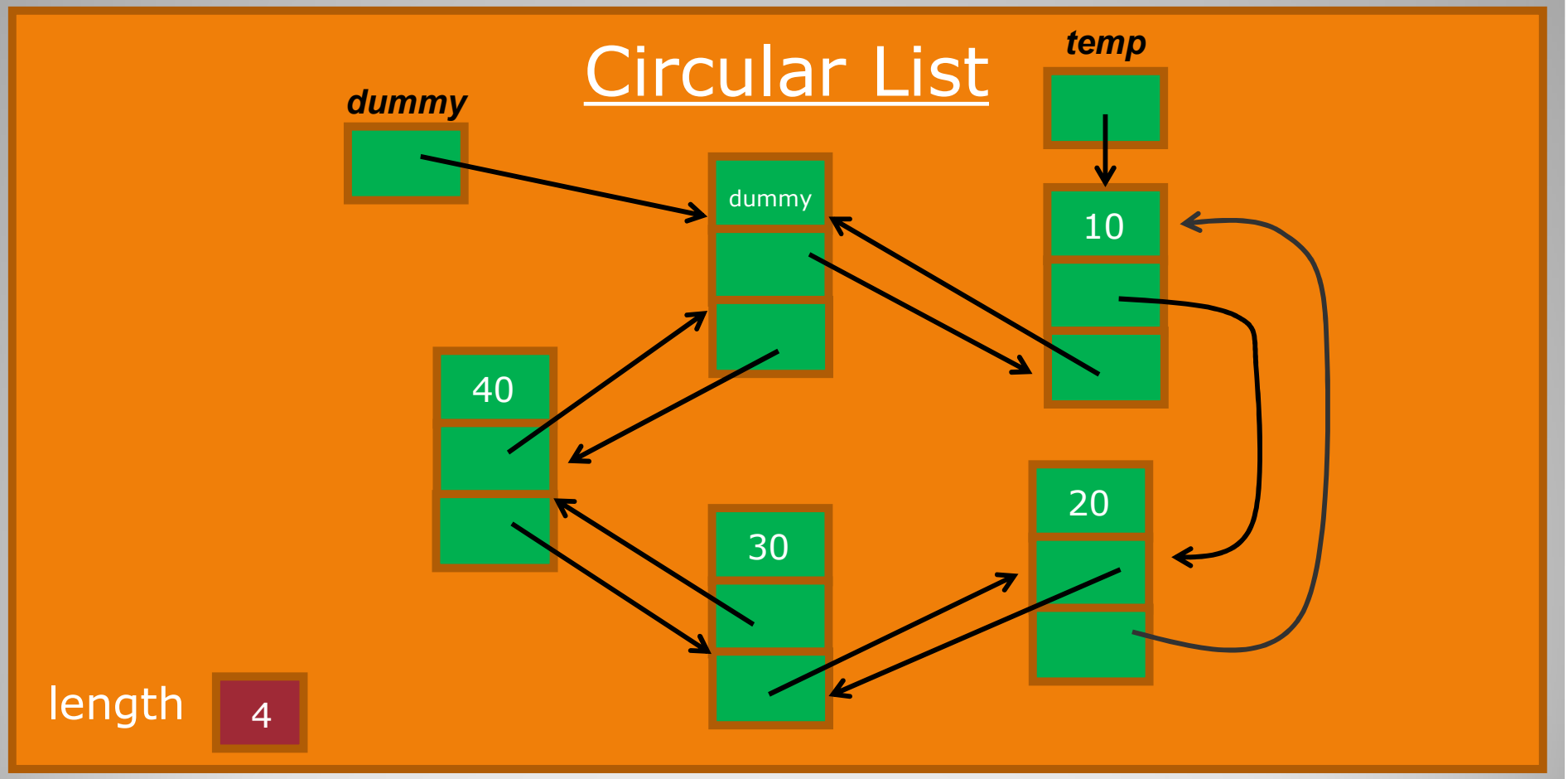
4. Set temp.next.previous to temp.



Circular -insertHead

cl.insertItem(10);

5. Increment the length of the list.



Circular -insertHead

insertTail Pseudocode

1. Create a new Node instance named temp (dynamically allocate).
2. Set the fields on the new Node.
   A. Set the data.
   B. Set the next pointer to dummy.
   C. Set the previous to dummy's previous.
3. Set temp.previous.next to temp (the new node).
4. Set temp.next.previous to temp (the new node).
5. Increment the length of the list.

Circular Doubly-linked List w/Dummy - insertTail

- How do we traverse a circular list from beginning to end?
- Where do we start?
- For example…

Circular Doubly-linked List w/Dummy - Traversal

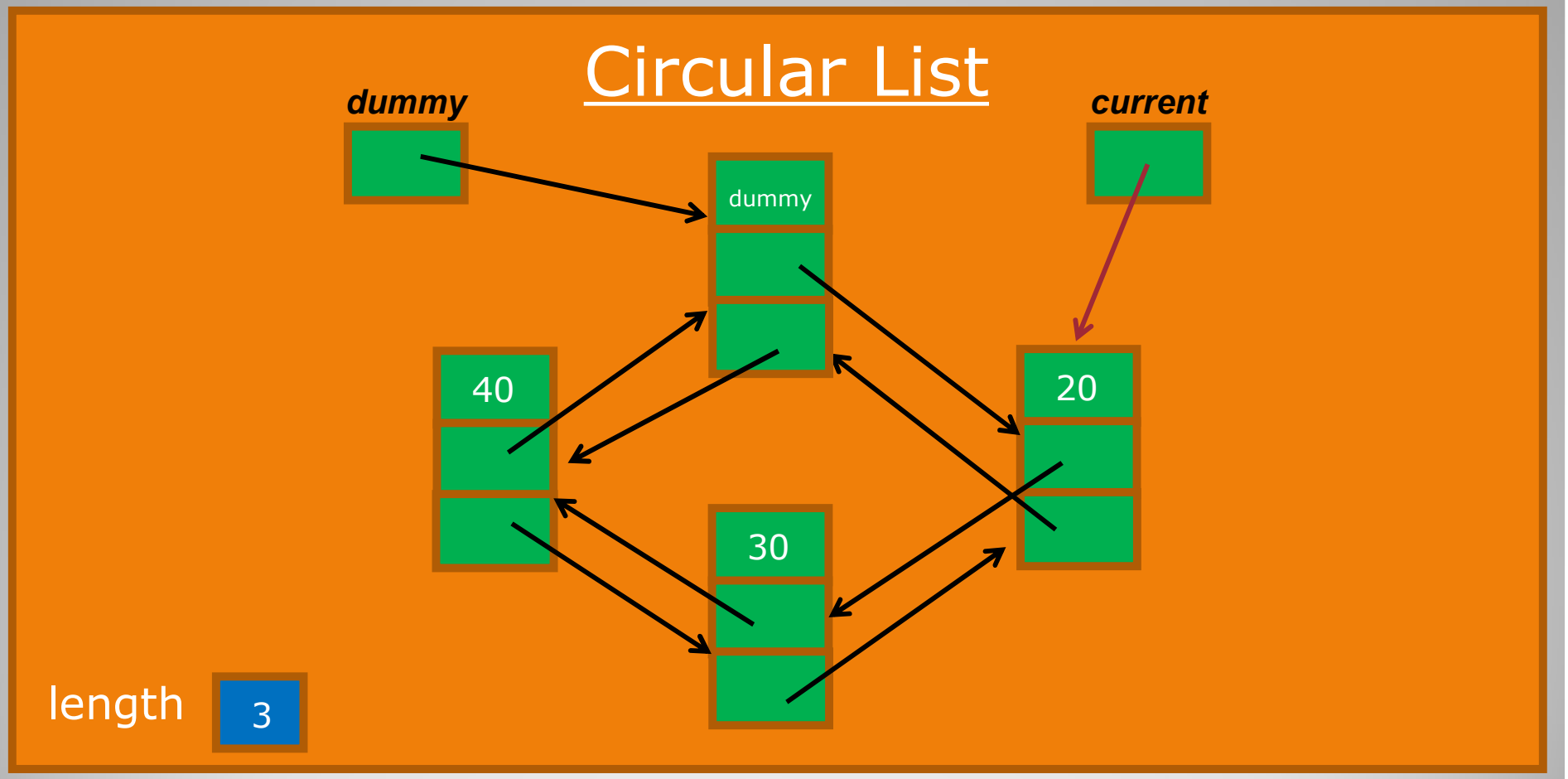
Traversal Pseudocode (beginning to end)

1. Set local variable current to the head.
2. While current not equal to dummy.
   A. Print data at current.
   B. Move current to next node.

Circular Doubly-linked List w/Dummy - Traversal

1. Set local variable current to the head (dummy.next)

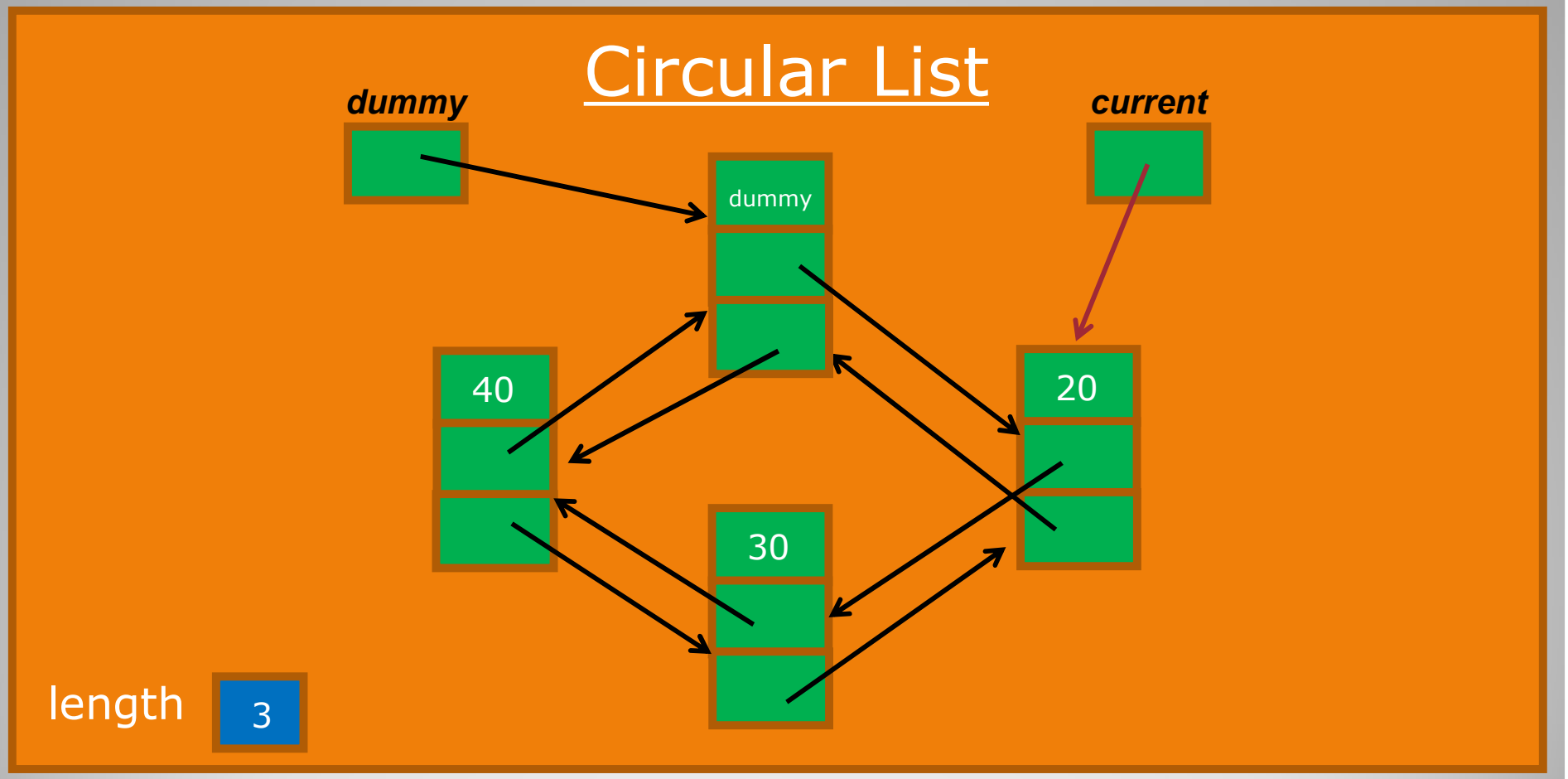
Output



Circular - Traversal

Output

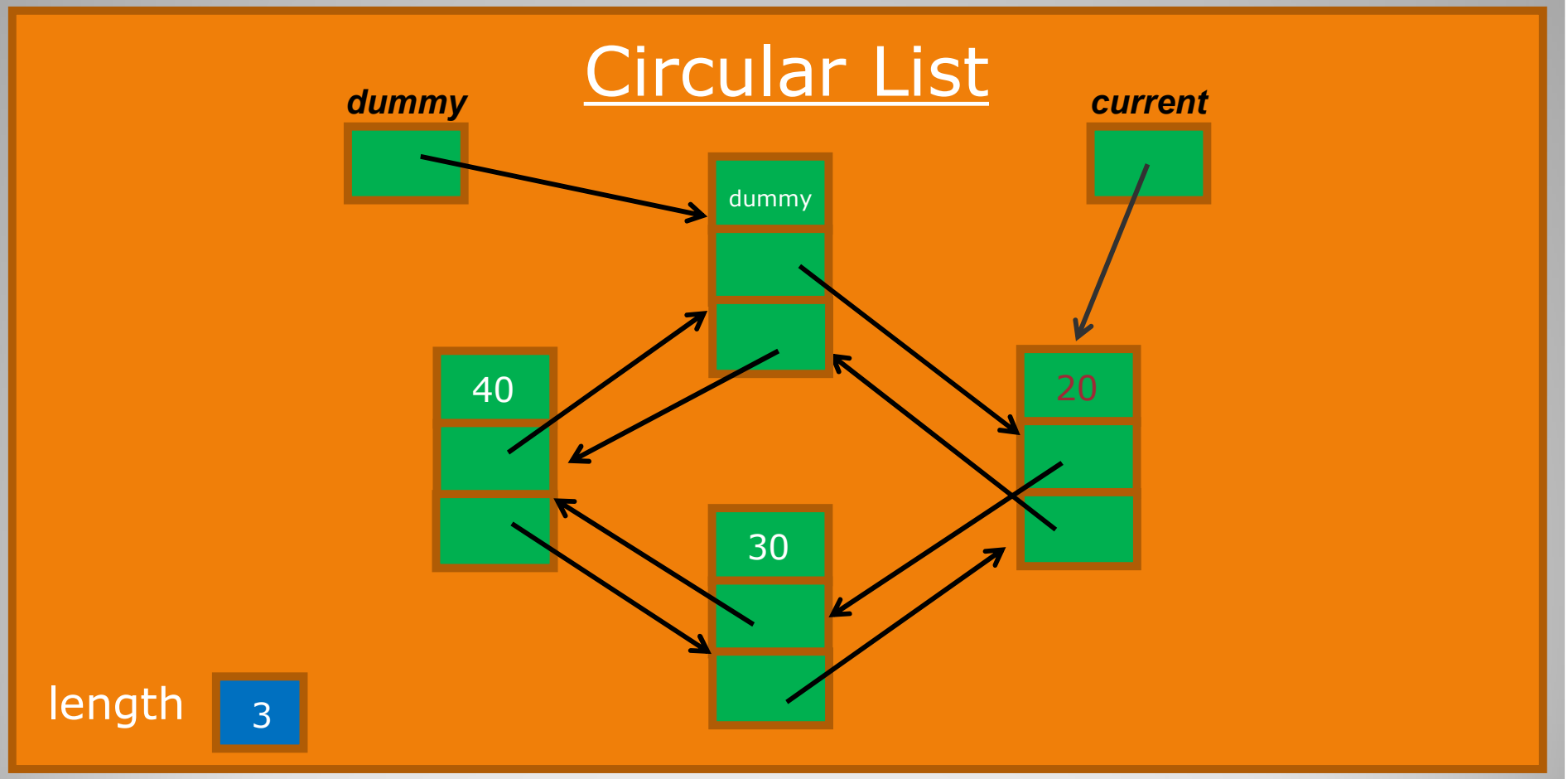
2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

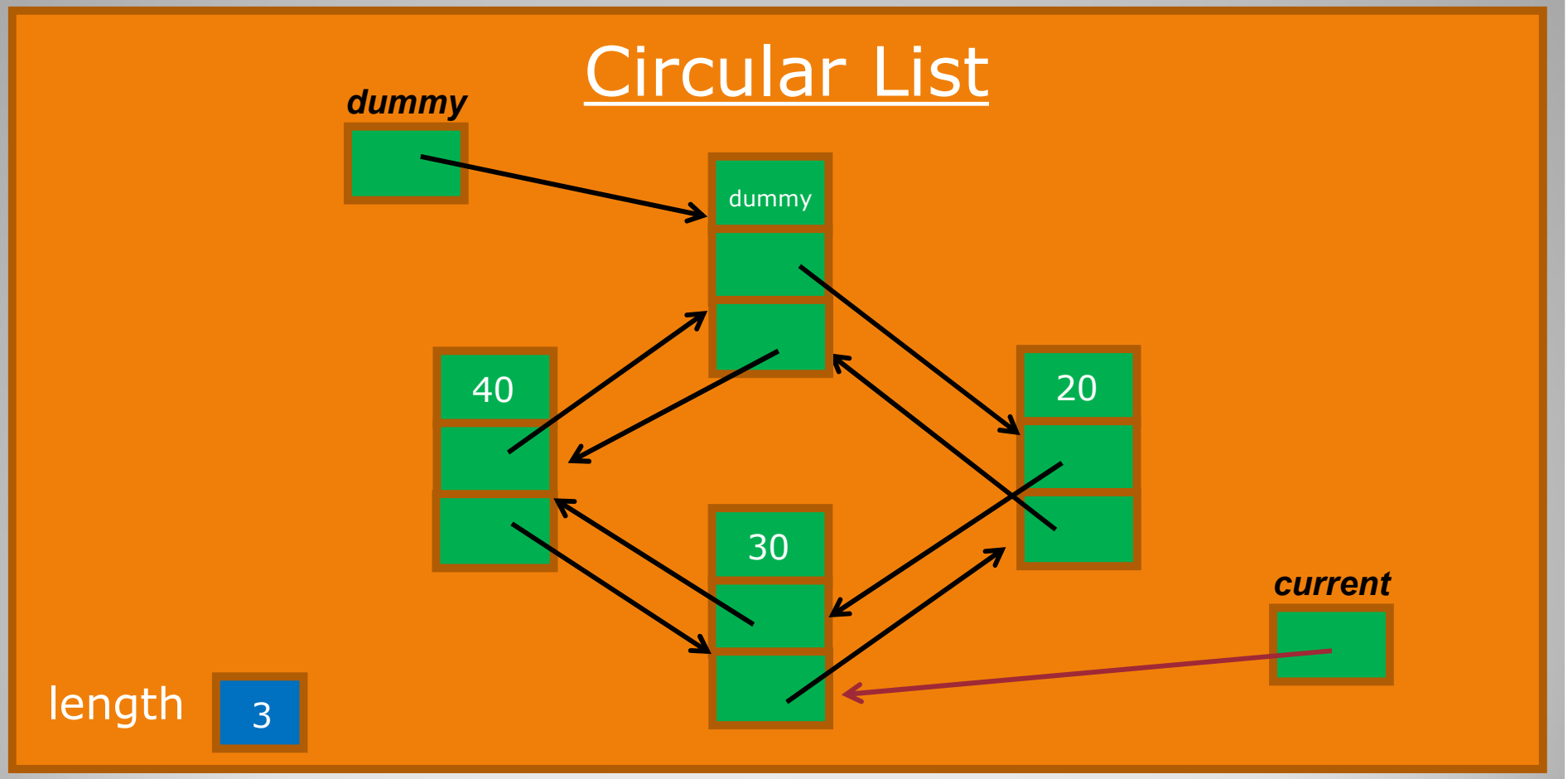
Output
20



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

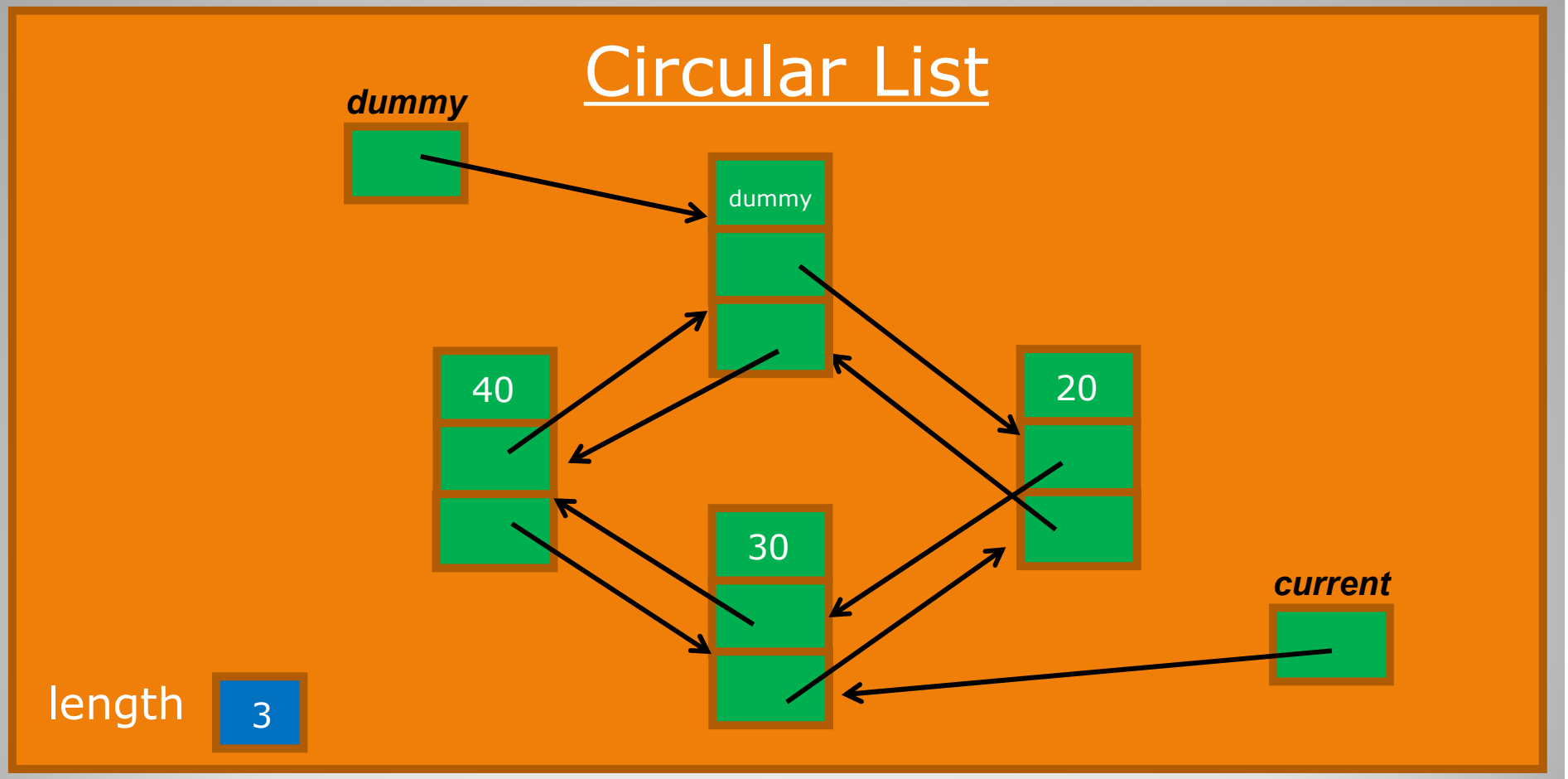
Output
20



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

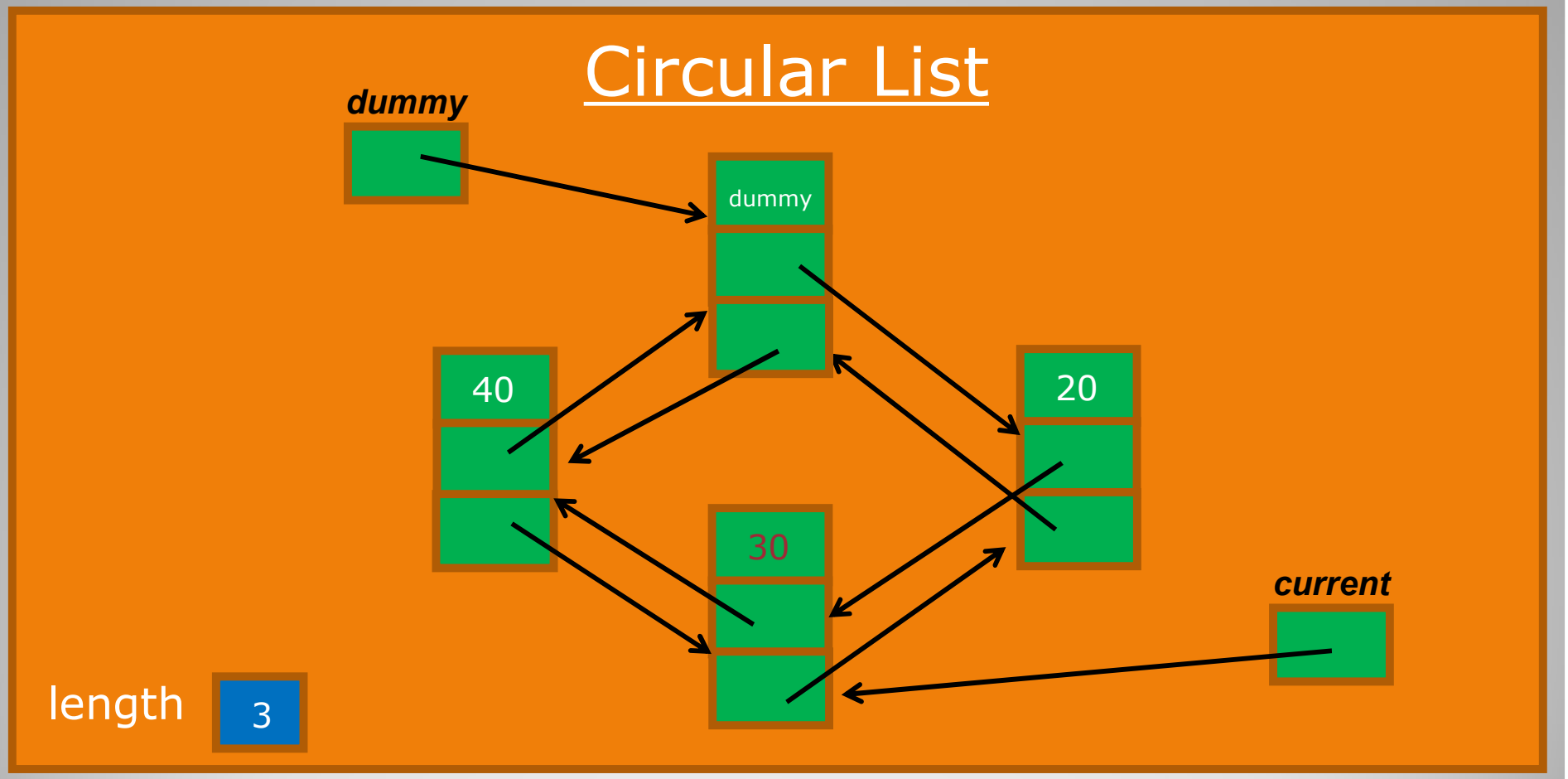
Output
20



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

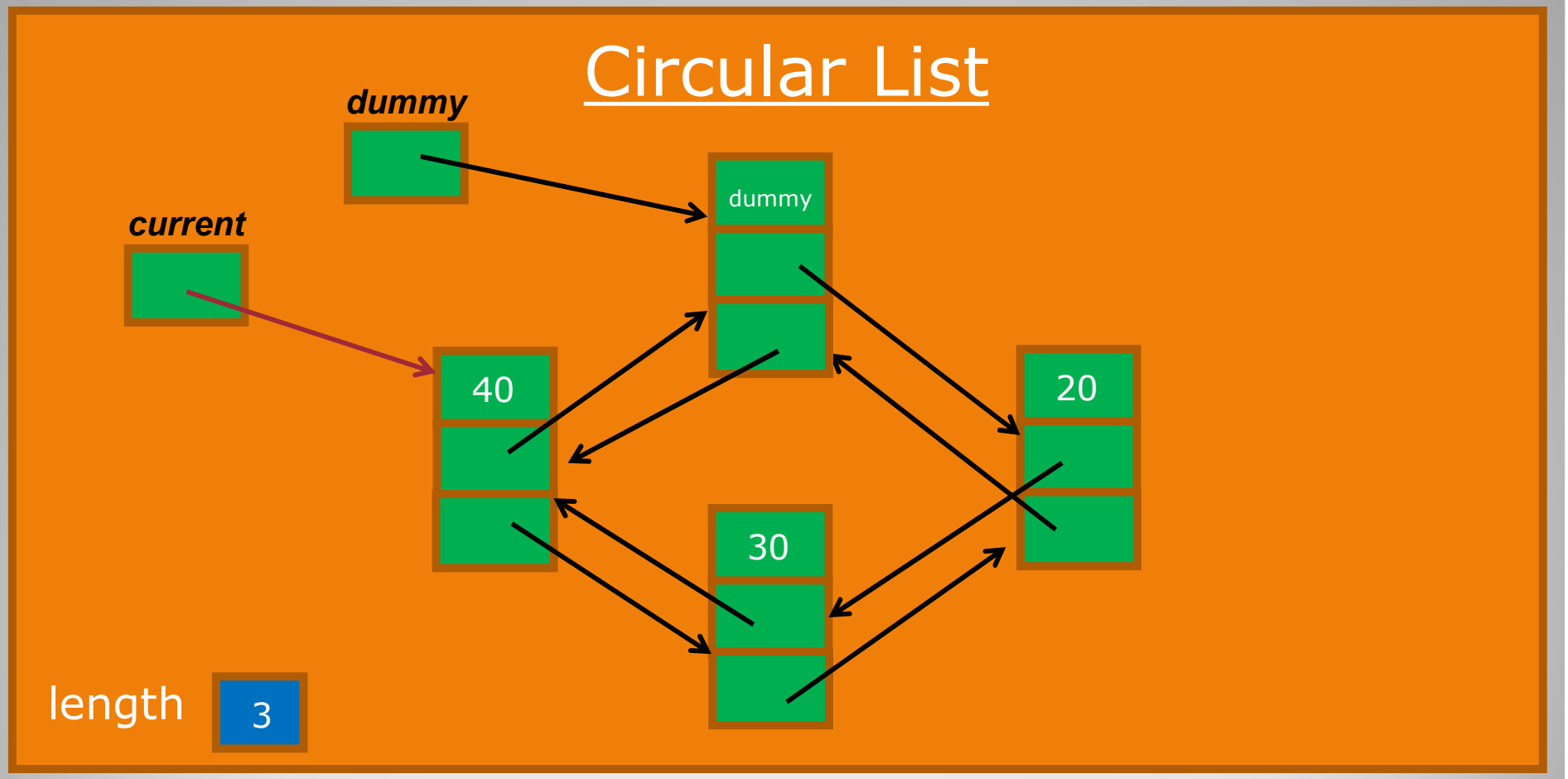
Output
20 30



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

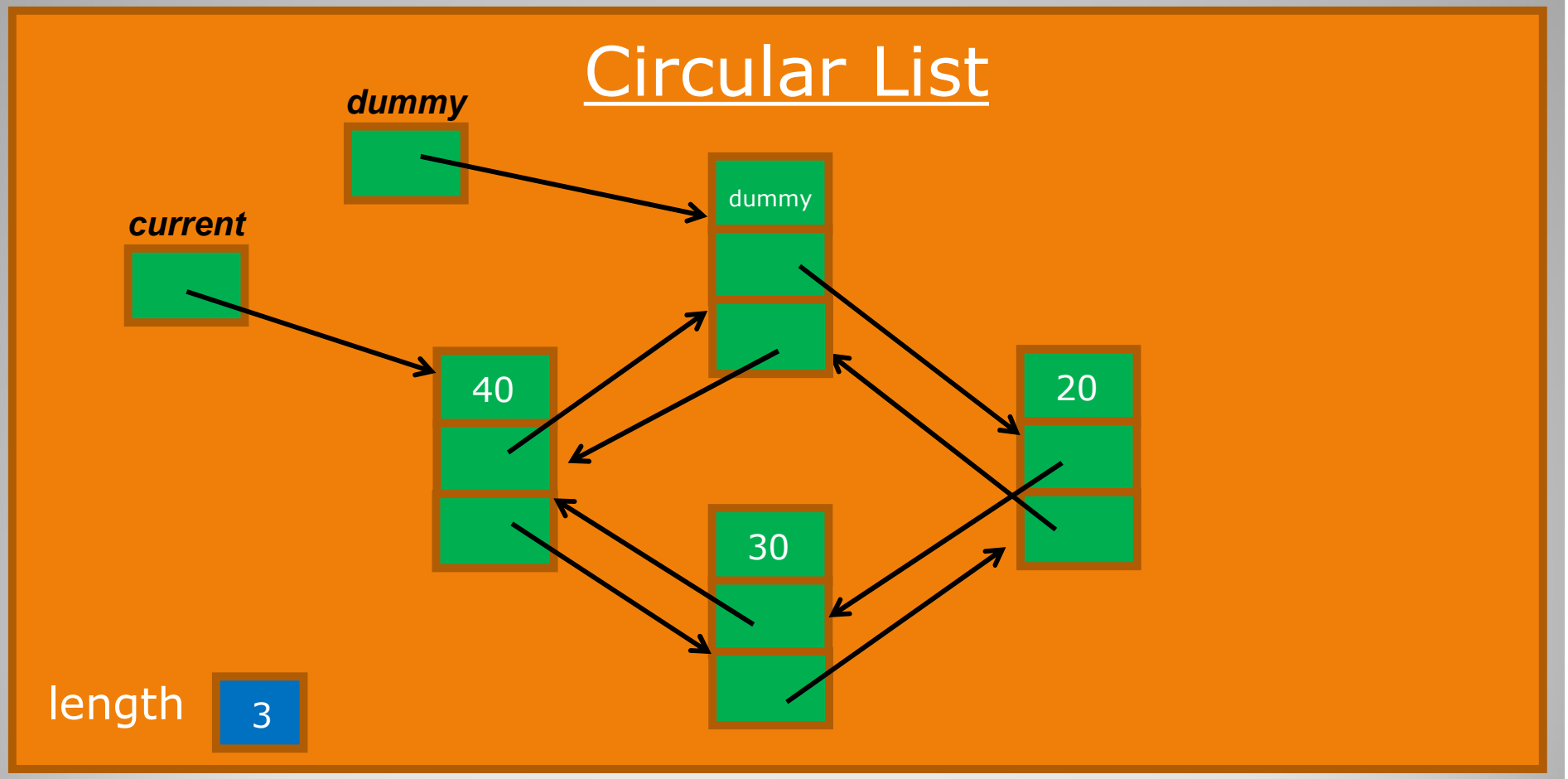
Output
20 30



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

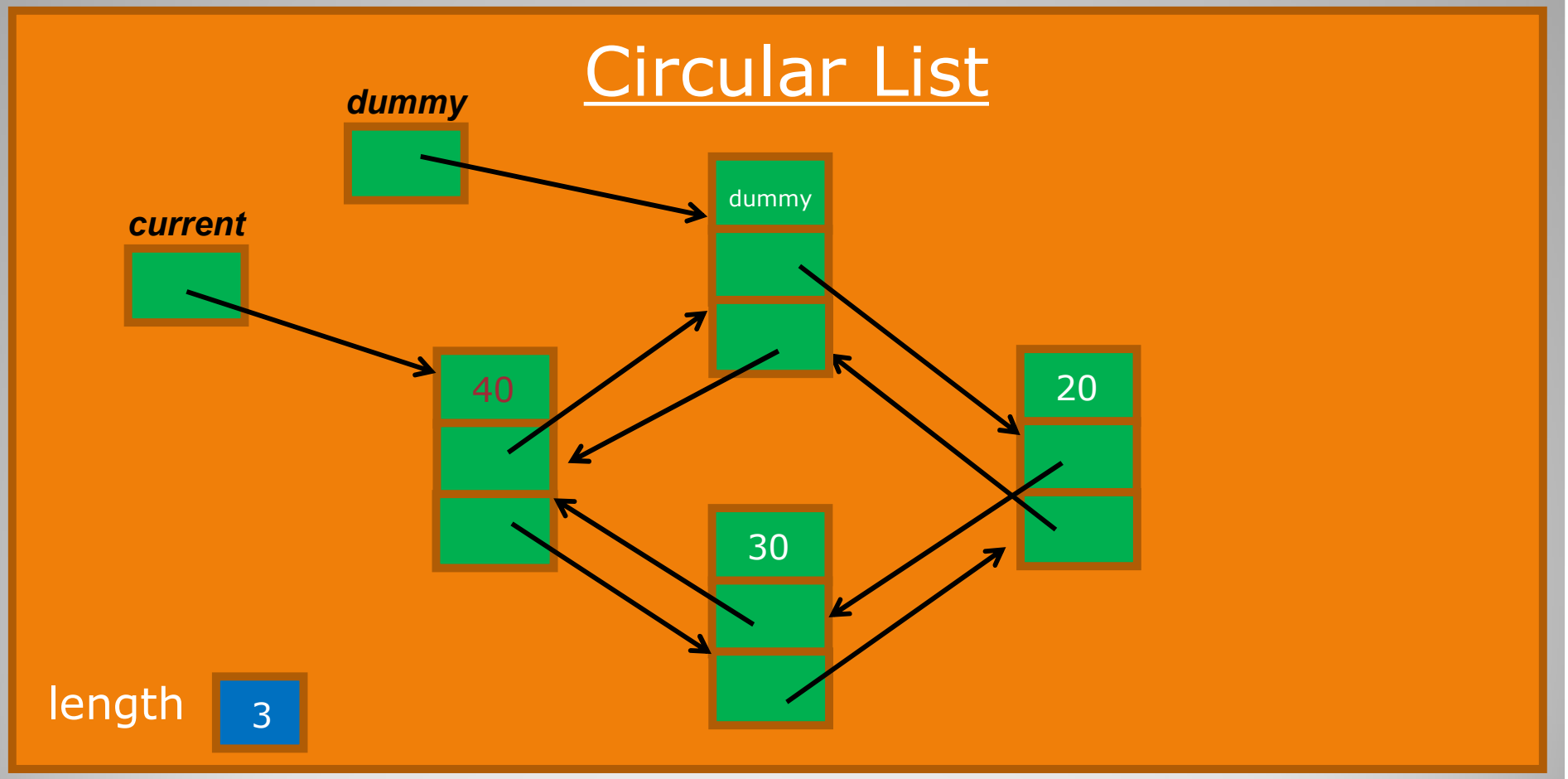
Output
20 30



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

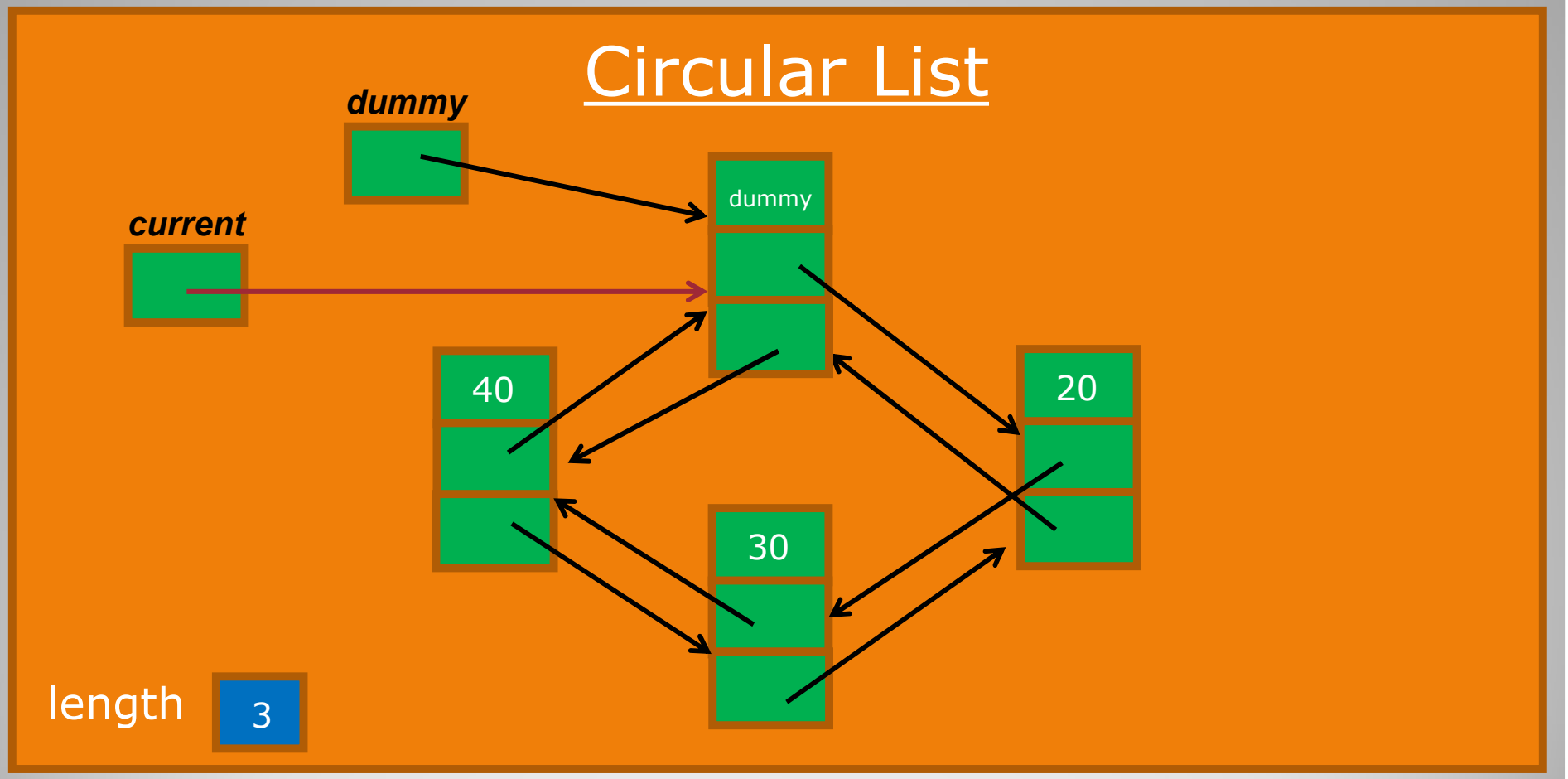
Output
20 30 40



Circular - Traversal

2. While current not equal to dummy (true so go in loop)
   A. Print data at current node
   B. Move current to next node

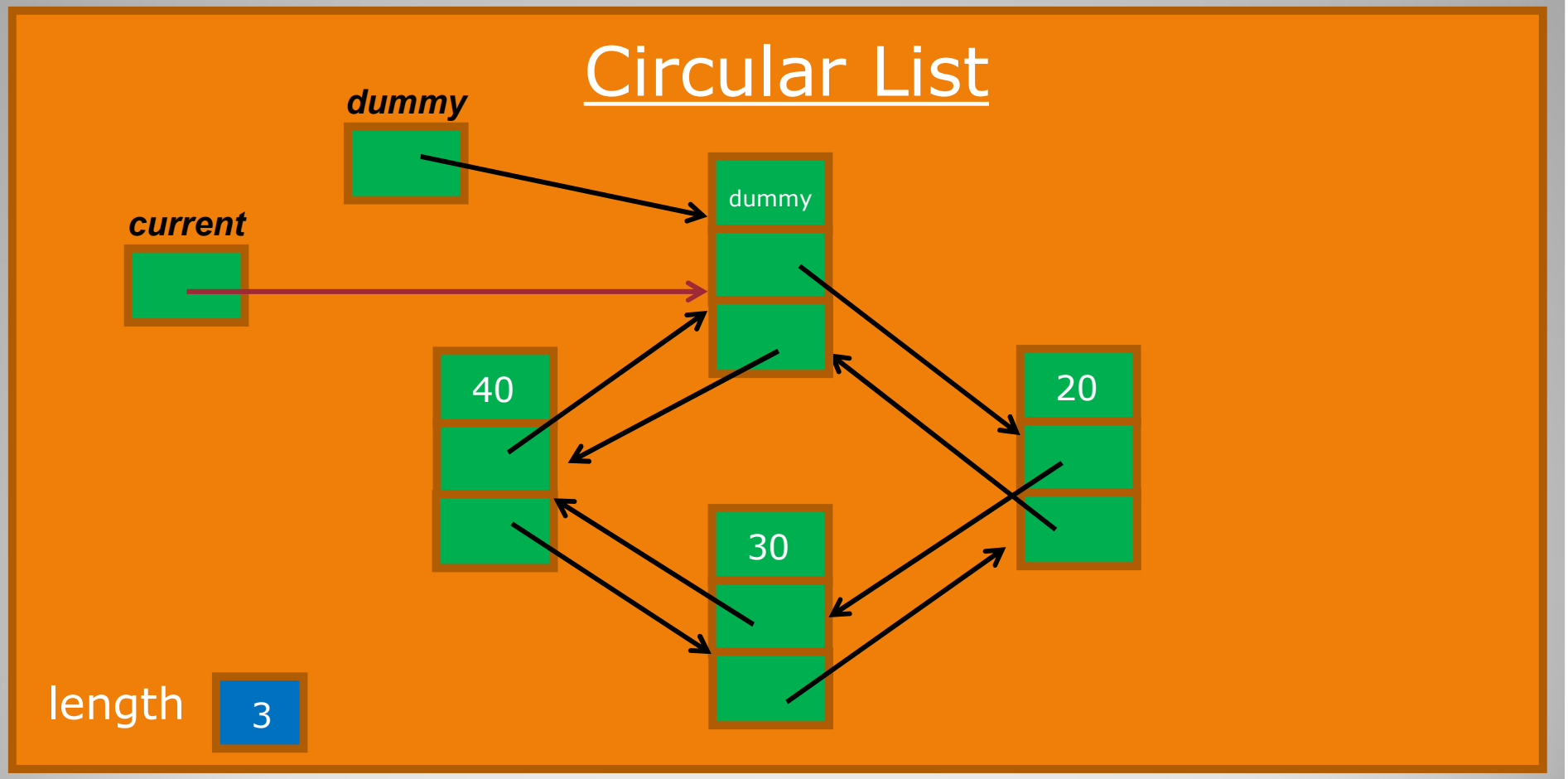
Output
20 30 40



Circular - Traversal

2. While current not equal to dummy (false so leave loop)
   A. Print data at current node
   B. Move current to next node

Output
20 30 40



Circular - Traversal

- **End of Slides**

End of Slides