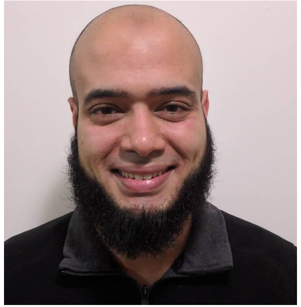
Data Structures

Display Nodes

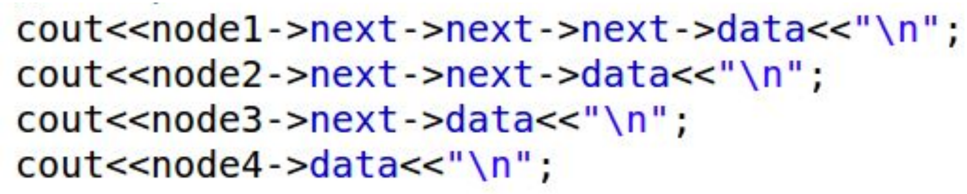
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

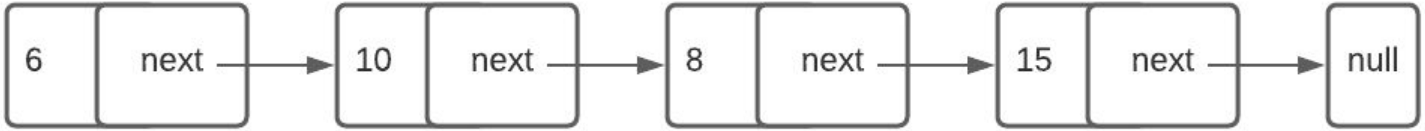


So far

- We learned how to create and do manual navigation to the items
- Let’s write a function that prints all values starting from the head (6 here)

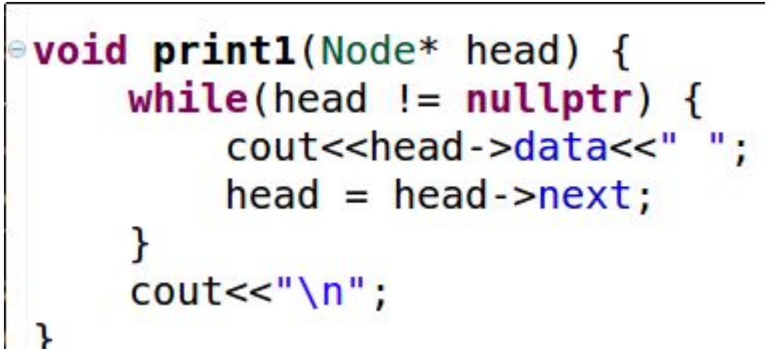


```
cout<<node1->next->next->next->data<<"\n";
cout<<node2->next->next->data<<"\n";
cout<<node3->next->data<<"\n";
cout<<node4->data<<"\n";
```

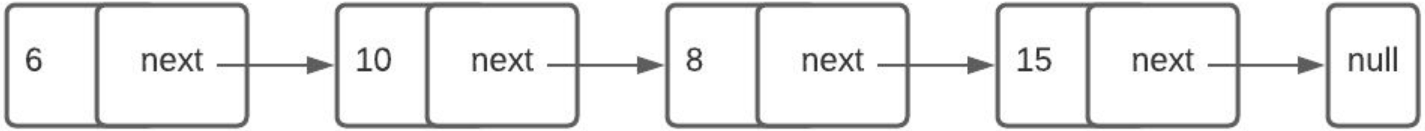


Printing Nodes Chain

- Let’s build over last code
- From main:
  - print1(node1);
- The output is: 6 10 8 15
- Take 10 minutes try to trace by yourself!

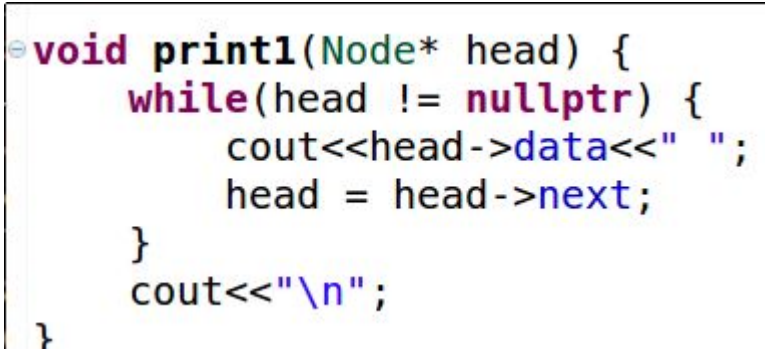


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

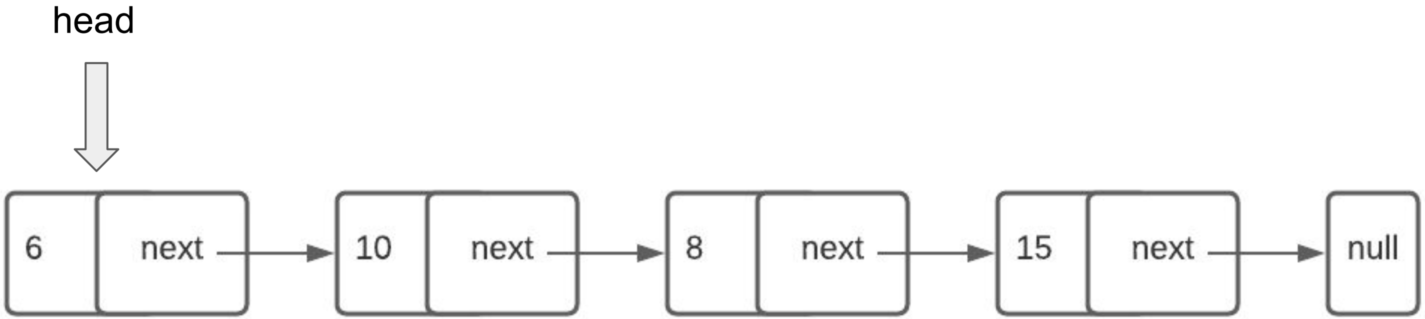


Printing Nodes Chain: Trace

- Initially head pointer is pointing to node 1
  - ○ Check is not null? No
  - ○ Print the value ⇒ 6
  - ○ What is head->next? node2
  - ○ Set head = node2

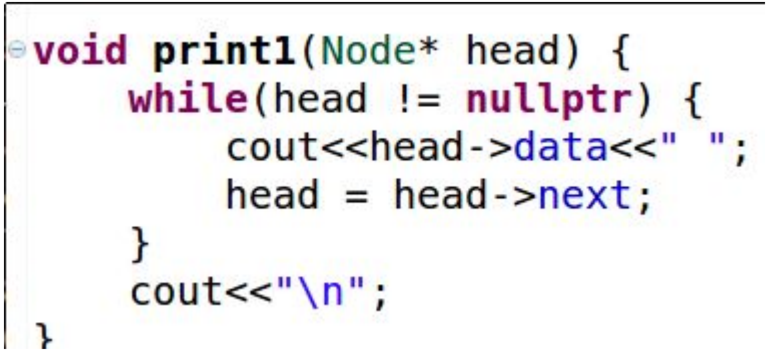


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

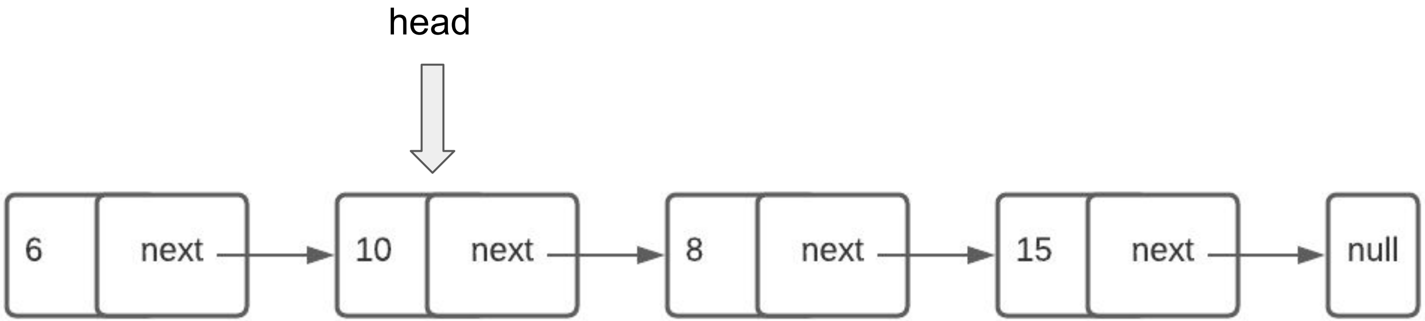


Printing Nodes Chain: Trace

- Now head pointer is pointing to node 2
    - ○ Check is not null? No
    - ○ Print the value ⇒ 10
    - ○ What is head->next? node3
    - ○ Set head = node3

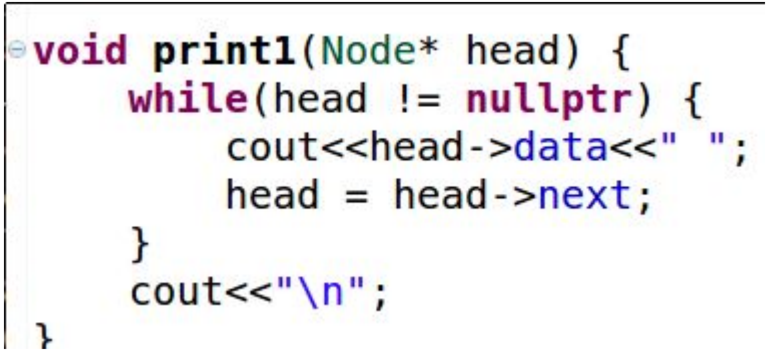


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

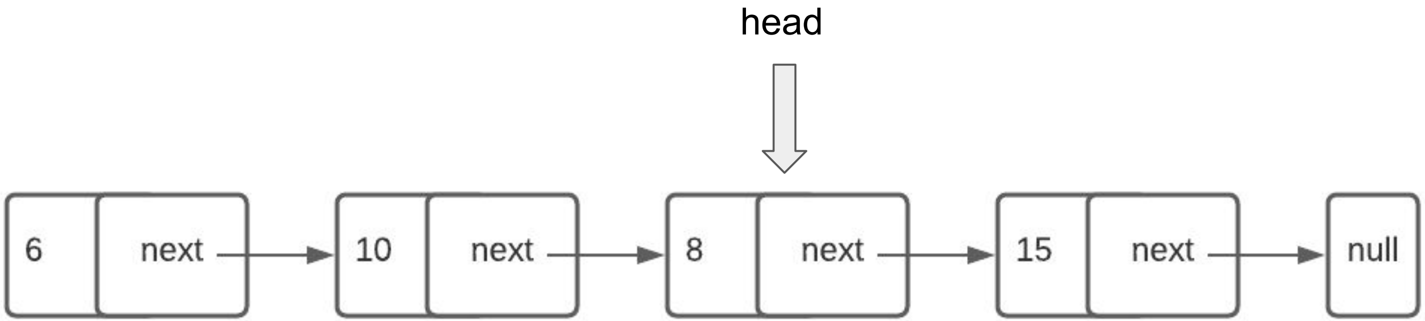


Printing Nodes Chain: Trace

- Now head pointer is pointing to node 3
  - Check is not null? No
  - Print the value ⇒ 8
  - What is head->next? node4
  - Set head = node4

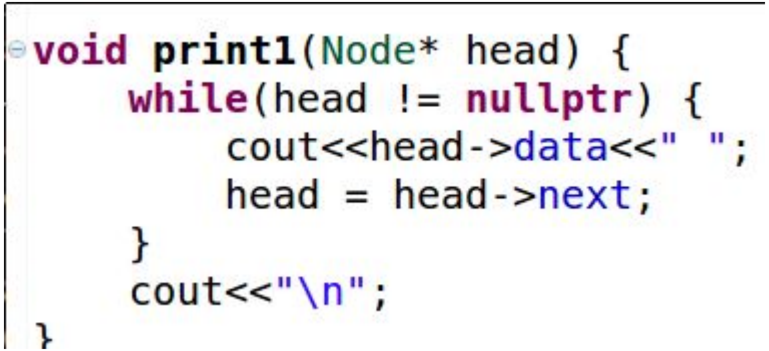


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

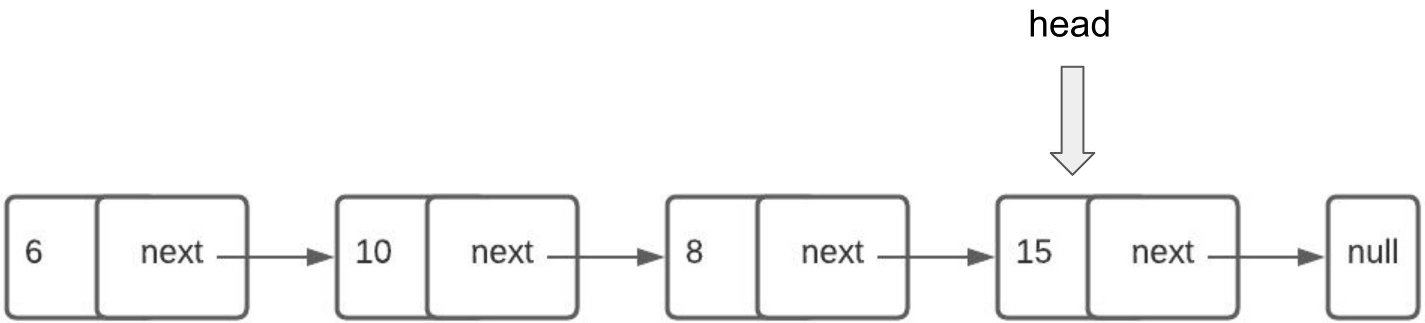


Printing Nodes Chain: Trace

- Now head pointer is pointing to node 4
  - ○ Check is not null? No
  - ○ Print the value ⇒ 15
  - ○ What is head->next? null
  - ○ Set head = null

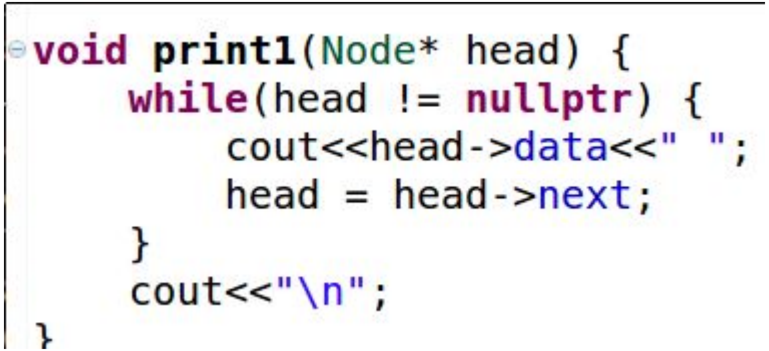


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

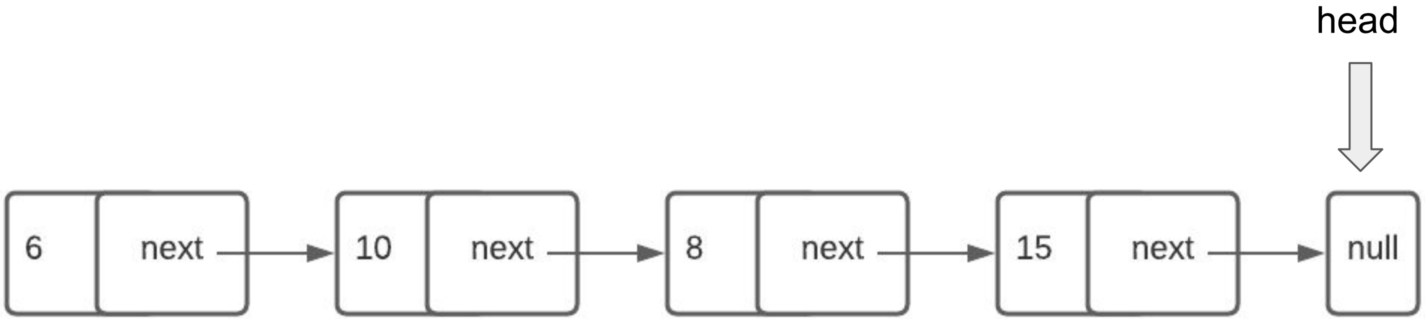


Printing Nodes Chain: Trace

- Now head pointer is pointing to null
  - Check is not null? YES stop
- This code is very fundamental
- Make sure of 100% understanding
- Your turn:
  - Can you rewrite this code to be recursive?

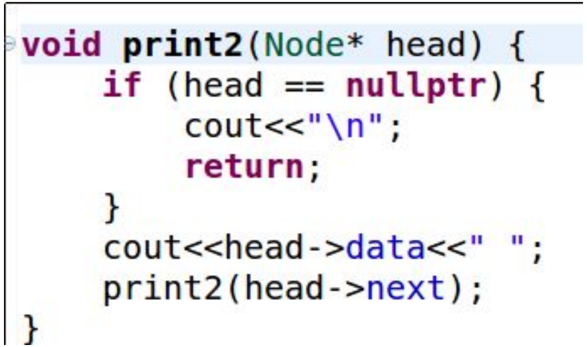


```
void print1(Node* head) {
    while(head != nullptr) {
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<"\n";
}
```

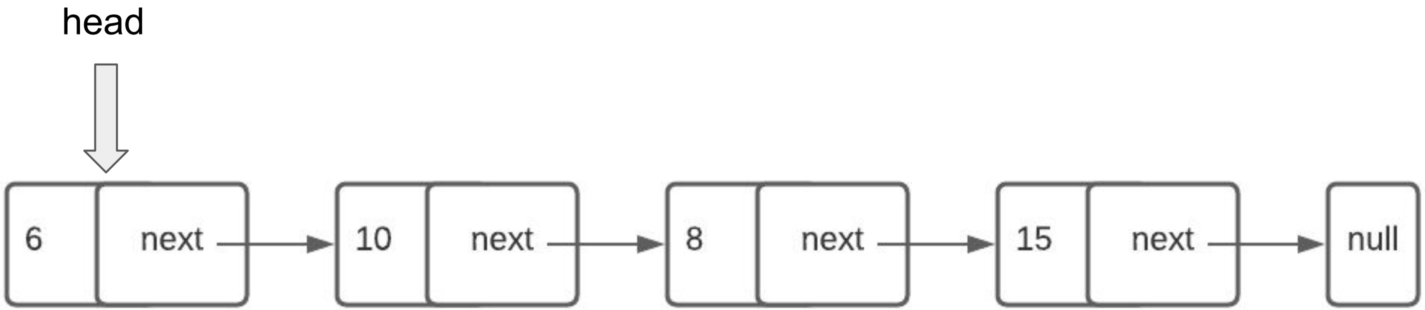


Printing Nodes Chain: Recursively

- This is exactly like printing an array recursively
  - Print
  - Call next array element (node->next)
- Similarly, try to print reversed
  - 15 8 10 6

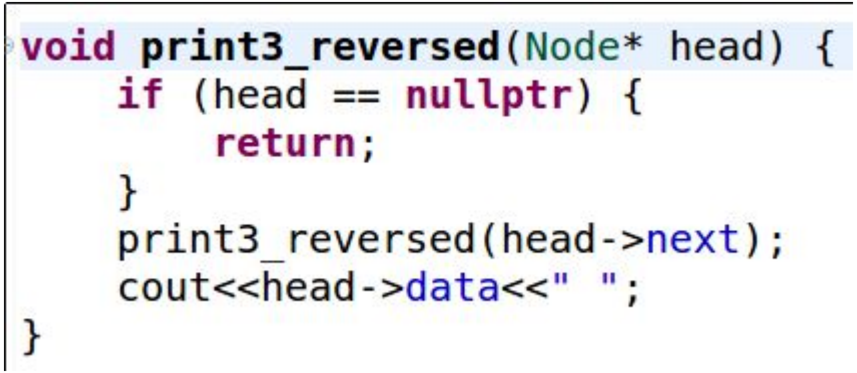


```
void print2(Node* head) {
    if (head == nullptr) {
        cout<<"\n";
        return;
    }
    cout<<head->data<<" ";
    print2(head->next);
}
```

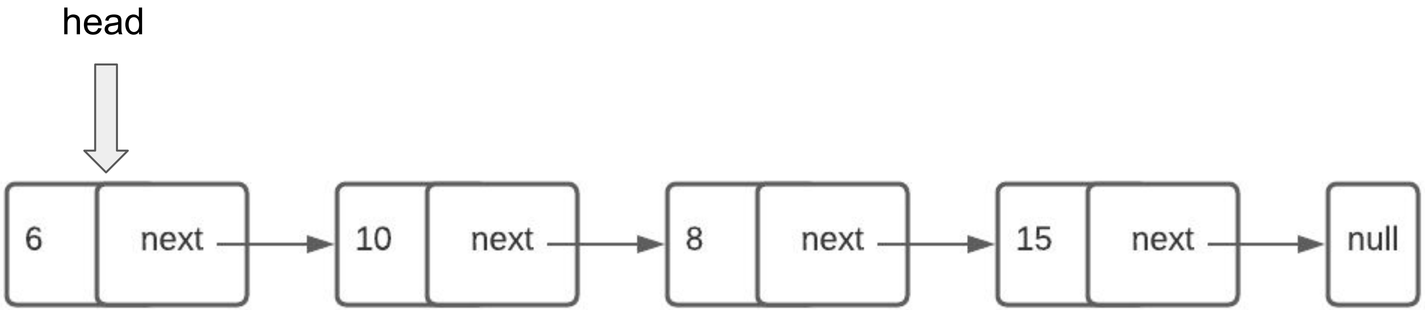


Printing Nodes Chain: Recursively

- The whole trick to call recursively first, then print
  - Then once we reach nullptr, we print



```
void print3_reversed(Node* head) {
    if (head == nullptr) {
        return;
    }
    print3_reversed(head->next);
    cout<<head->data<<" ";
}
```



Your turn

- Understand the code very well
- Play with the code
- Try to implement the following ideas
  - Function find(value) that **searches** for a node with the given value
    - If found it, return the node
    - Otherwise return null
  - Rewrite the print1 function using **for loop** instead of while loop

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”