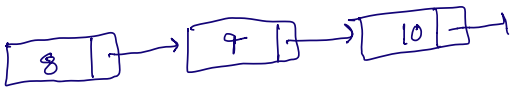
Singly Linked List

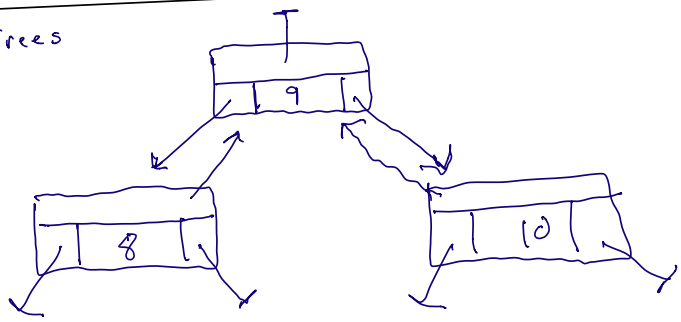



```
struct Node {
  Node* next;
  int val;
};
```

Doubly Linked List


```
struct Node {
  Node* next;
  Node* prev;
  int val;
};
```

Trees




```
struct Node {
  Node* left;
  Node* right;
  Node* parent;
  int val;
};
```

```
template <typename T>
class SinglyLinkedList {
public:
    SinglyLinkedList();
    SinglyLinkedList(T);
    void push_back(T);
    void insert_at(T, unsigned int);

private:
    Node<T> * head;
    Node<T> * tail;
    unsigned int sz;
};
```

```
template <typename T>
SinglyLinkedList<T>::SinglyLinkedList() :
        head{nullptr},
        tail{nullptr},
        sz(0)
{}
```

```
template <typename T>
SinglyLinkedList<T>::SinglyLinkedList(T data) :
        head{nullptr},
        tail{nullptr},
        sz(0)
{
    head = new Node<T>(data);
    tail = head;
    sz = 1;
}
```

```
template <typename T>
struct Node {
    T data;
    Node<T> * next;
    Node(T data) : data(data), next(nullptr) {}
};
```

SinglyLinkedList<int> sll;

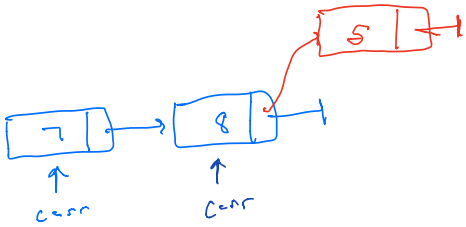
head →

SinglyLinkedList<int> sll2(7);

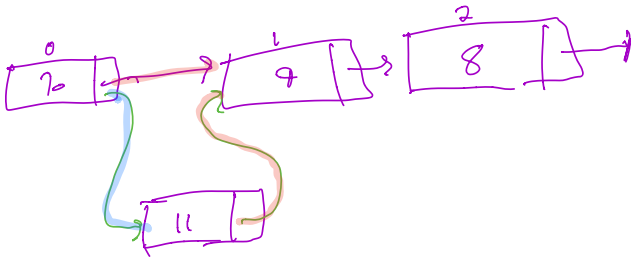
head → [7] →

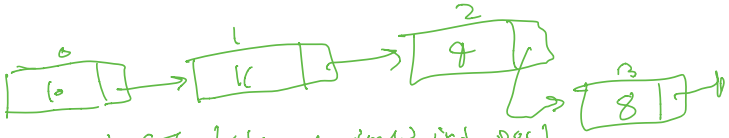
```
template <typename T>
void SinglyLinkedList<T>::push_back(T data)
{
    if (head == nullptr){
        head = new Node<T>(data);
        return;
    }
    Node<T> * curr = head;
    while (curr->next != nullptr)
        curr = curr->next;

    curr->next = new Node<T>(data);
    size++;
}
```



```
template <typename T>
void SinglyLinkedList<T>::push_back(T data)
{
    if (head == nullptr) {
        head = new Node<T>(data);
        tail = head; size=1;
        return;
    }
    Node<T>* n = new Node<T>(data);
    tail->next = n;
    tail = n;
    size=1;
}
```






```
template <typename T>
void SinglyLinkedList<T>::insert_at(T data, unsigned int pos)
{
    if (pos >= sz)
        throw RuntimeError("Out of bounds");
    if (pos == 0) {
        Node<T>* n = new Node<T>(data);
        n->next = head; head = n; sz++;
        return;
    }
    Node<T>* curr = head;
    for (int i=0; i<pos; ++i)
        curr = curr->next;
    Node<T>* n = new Node<T>(data);
    n->next = curr->next;
    curr->next = n;
    sz++;
}
```

// Still error when does this on the
// fly in class. Loop should be
// for (int i=0; i<pos-1; ++i)
// to get one before insertion position
// version in case tests you to position pos,
// but we want to be one before! :)

```
template <typename T>
SinglyLinkedList<T>::SinglyLinkedList(const SinglyLinkedList<T>&
                                      rhs) :
    head(nullptr),
    tail(nullptr),
    sz(0)
{
    Node<T>* iter_rhs = rhs.head;
    while (iter_rhs != nullptr)
    {
        push_back(iter_rhs->data);
        iter_rhs = iter_rhs->next;
    }
}
```