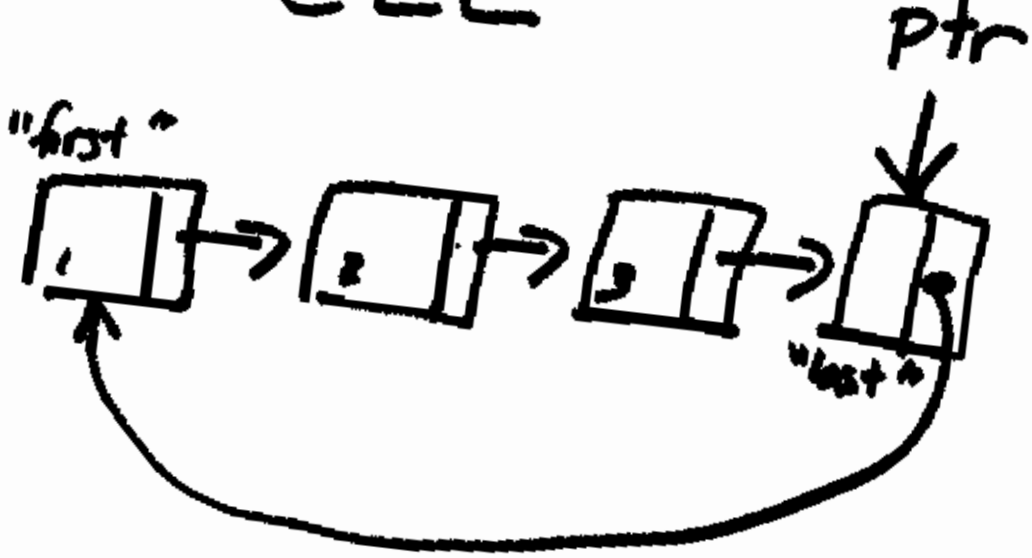
Circular Linked List

CLL



cout << ptr->next->data

"first"

ptr  if List is Empty

Display all LLL

```
node * temp = head;
while ( temp )
{
        cout << temp -> data
             << endl;
        temp = temp -> next;
}
```

1 ✓ ✓

(Also Display all for DLL)

Display-all CLL

1) Special case

```
if (!ptr) return;
```

2)

```
node * temp = ptr->next;
do
{
    cout << temp->data
         << endl;
    temp = temp->next;
} while (temp != ptr->next);
```

1 2

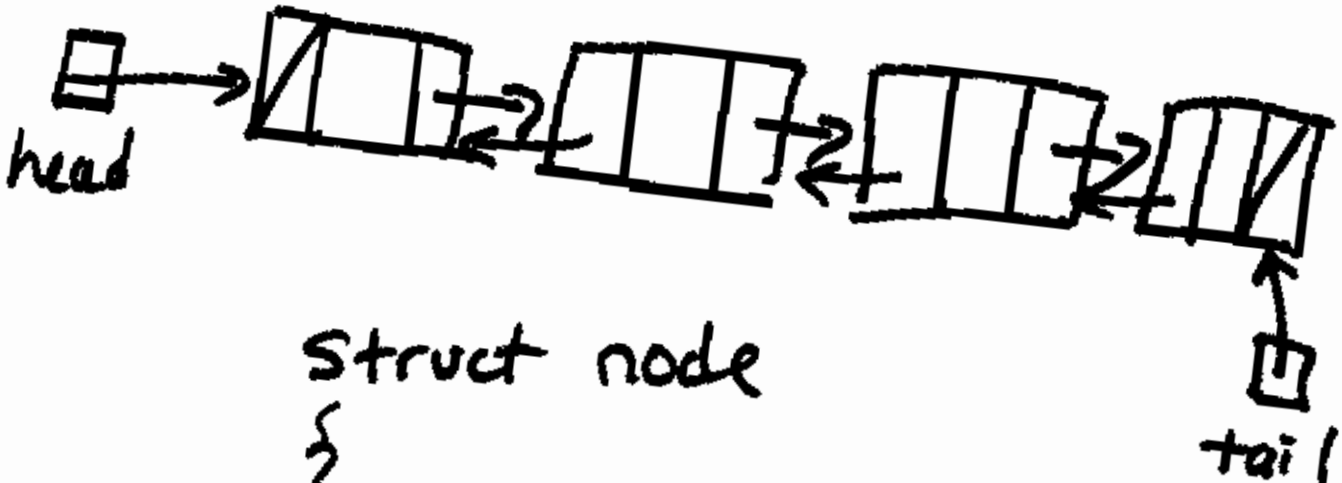
~ 6 op. & fetches

Shortcut for CLL

```
node * hold = ptr->next;
ptr->next = NULL;
temp = hold;
while (temp)
{
        cout << temp->data
             << endl;
        temp = temp->next;
}
ptr->next = hold;
```

Doubly Linked List

DLL




```
Struct node
{
    student data;
    node * next;
    node * previous;
};
```

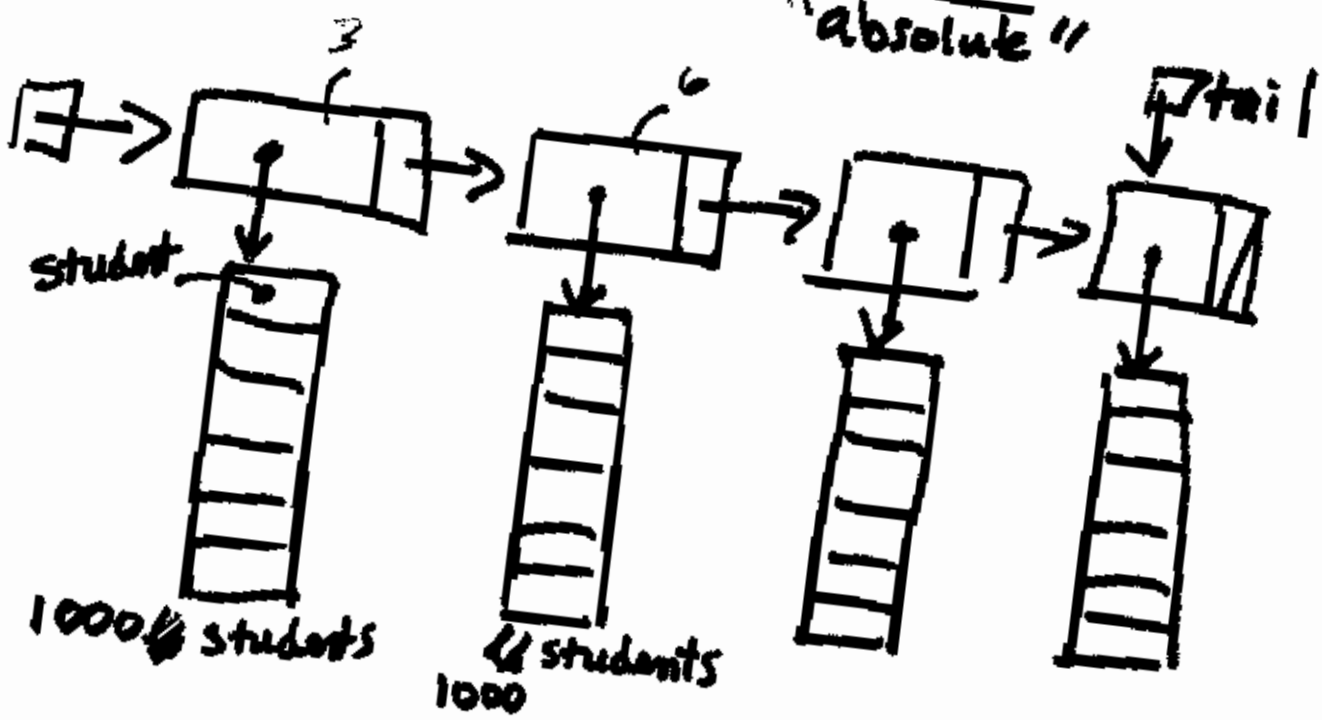
10,000 Nodes we have:

20,000 Adresses + 2

DLL Display Backward

```
node * temp = tail;
while (temp)
{
        cout << temp->data
             << endl;
        temp = temp->previous;
}
```

LLL of Arrays




```
Struct node
{
        Student array[SIZE = 1000]; //statically
        node * next;
};
```

```
Struct node
{
        student * array; //dynamically
        node * next;
};
```

- First Node

if dynamic

```
if(!head)
{   head = new node;
    head->array = new student [size];
    head->next = NULL;
}
```

tail = head;

- Saving data


```
head->array[0].copystudent(astudent);
```

OR

```
head->array[index].copystudent(...);
++index;
```

- when adding

```
if (head && index < size)
{   head->array[index]. . . . .
    ++index;
}
else if ( index >= size)
{   index = 0;
```

tail & current being same {

```
Current = tail -> next.
current -> array = new student[
current -> Next = NULL;
current -> array[index]. copy ...
++index;
}
```

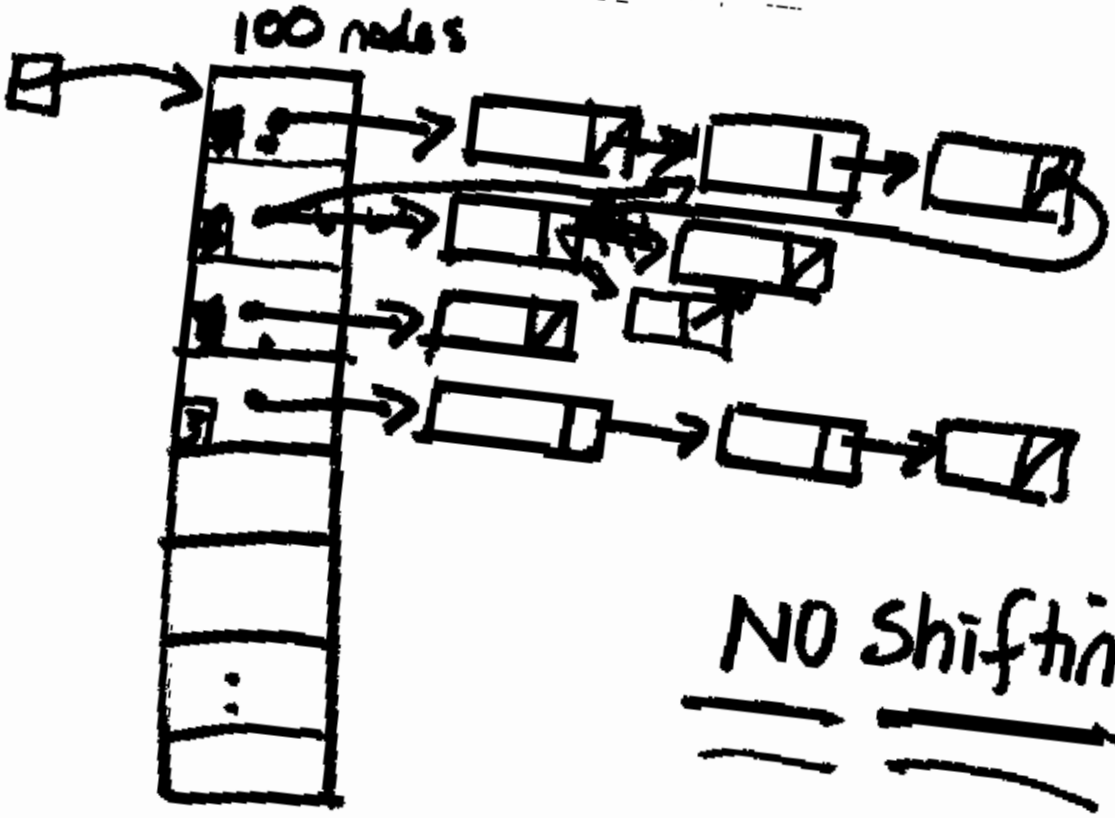
Go To Position 300 (Absolute)

How far to Traverse = 300/size

which element = 300 % size

Array of Linear Linked List

"Relative"



NO Shifting!

what about ?