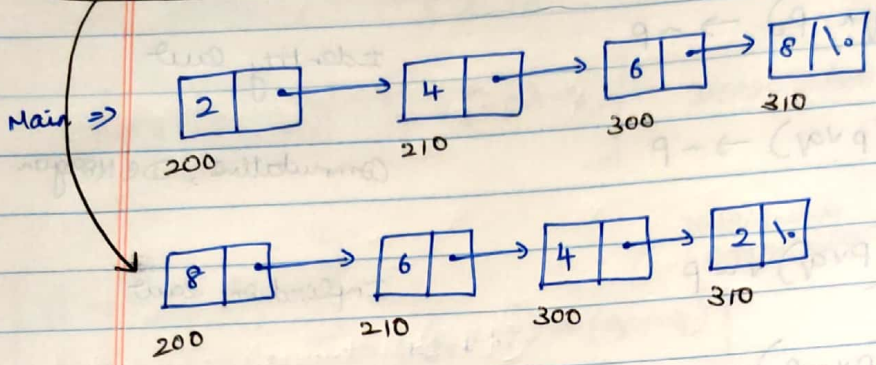
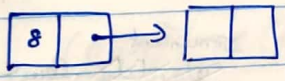
Reversing a Linked list

- Reversing the elements
- Reversing the links

Reversing the elements:

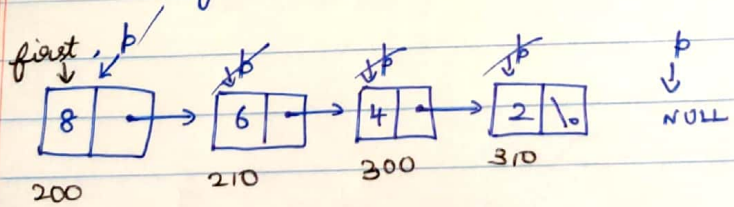


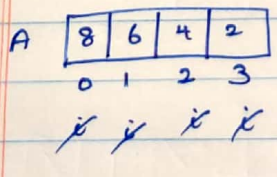
Reversing the links:



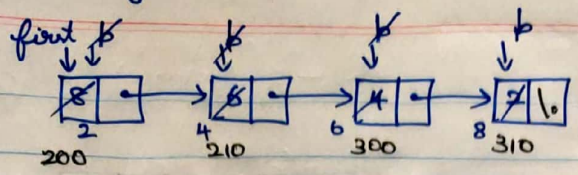
Reversing the elements

1) Copy all the list elements in an array, while returning back fill the array elements in the linked list.





then bring p => first, and decrement the count by simultaneously storing in the linked list



A | 8 | 6 | 4 | 2 |
0 1 2 3

i i i i i i

Code:

```
void Reverse (struct Node *ptr)
{
    struct Node *qtr = ptr;
int *A = (int *) malloc (sizeof (int) * count (ptr))
while (ptr != NULL)
{ ptr->data = A[i];      A[i++] = ptr->data;
  ptr = ptr->next;
}
ptr = qtr;
i--;
while (ptr != NULL)
{
  ptr->data = A[i--];
  ptr = ptr->next;
}
}
```

Copying the linked list elements to array

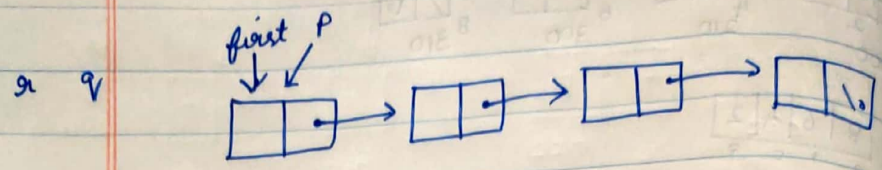
O(n)

Copying the array elements to linked list

O(n)

O(2n) => O(n)

Reversing using slide pointer



① r q p

② r q p

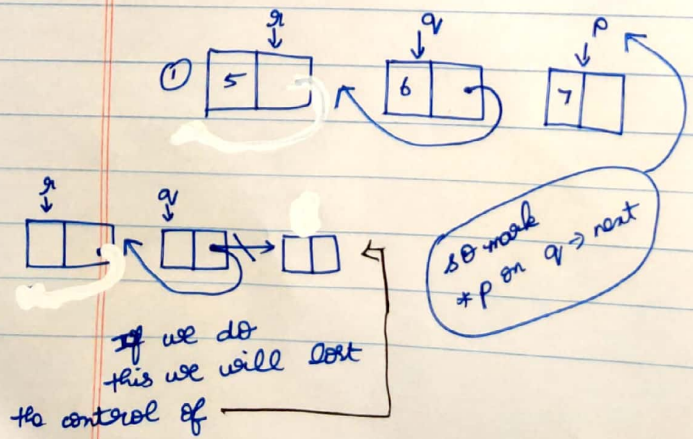
③ r q p

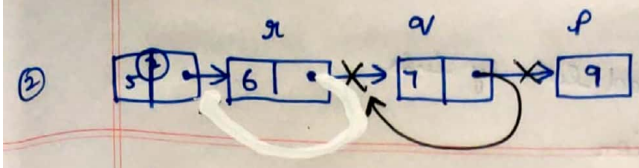
```
void Reverse2 (struct Node *p)
{
    struct Node *q = NULL, *r = NULL;
    while (p != NULL)
    {
        r = q;
        q = p;
        p = p->next;
        q->next = r;
    }
    first = q;
}
```

sliding pointers

for linking

⇒ After all the steps q should be our first pointer

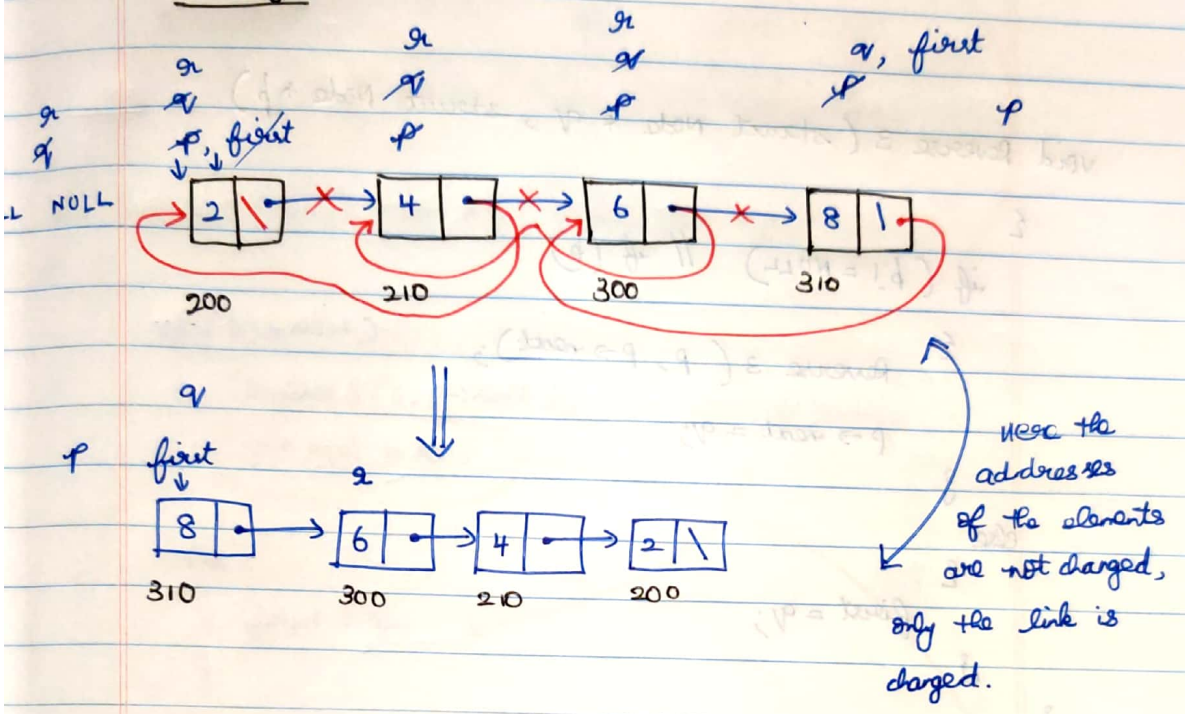




② So i) Slide all the three pointers.

ii) Reverse the q's link.

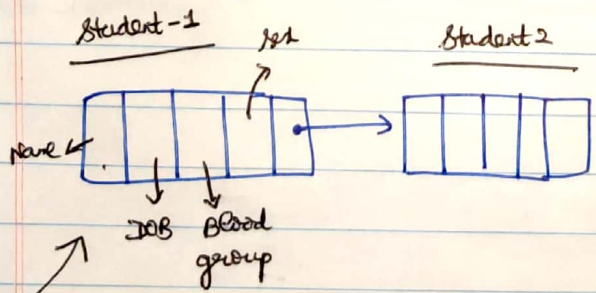
Testing



Note:

* We can reverse a linked list by ~~only~~ reversing the elements as well as reversing the links.

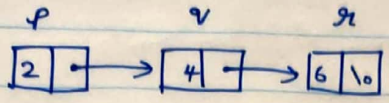
* But reversing the elements ⇒ Not professed Why?



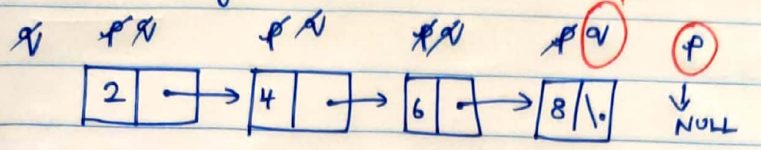
Here if we want to reverse the linked list by reversing the elements, we have to swap list 1 elements to other list, our case having only numbers so no problem but real time problems will be like this ⇒ So Not advisable Reversing the links

Reversing a linked list by links using recursion

While calling it requires, 3 pointers (i.e)

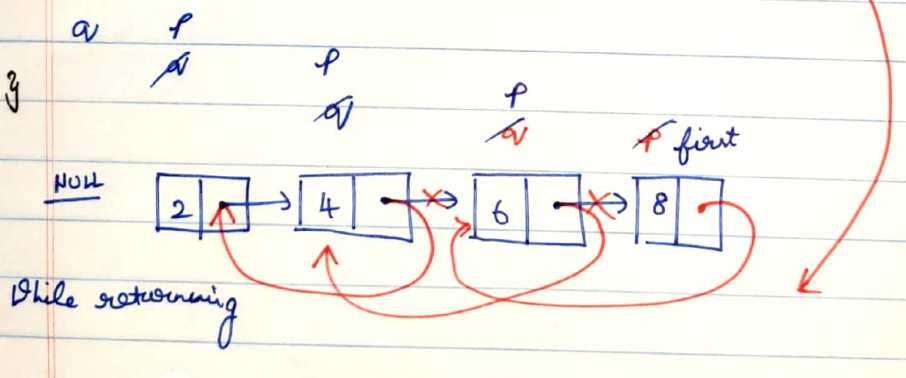


Do while returning time




```
void Reverse 3 (struct Node *q, Struct Node *p)
{
  if (p != NULL)
  { Reverse 3 (p, p->next);
    p-> next = q;
  }
  else
  { first = q;
  }
}
```

When p is NULL



While returning