

Some students have a problem in understanding the difference between `while(p!=null)` and `while(p.link!=null)`, so I have written this article with full explanation. Please go through it, and feel free to post in the Q&A forum, if anything is not clear.

FAQ 1 :

What is the difference between

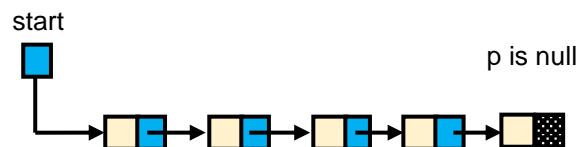
`while(p!=null)` ,

`while(p.link!=null)` and

`while(p.link.link!=null)` in linked lists?

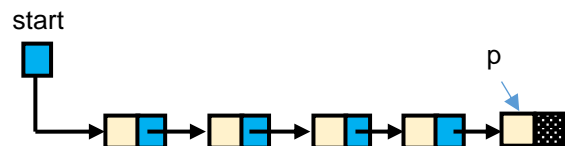
When to use which one?

```
p = start;
while(p!=null)
{ .....
  p = p.link; }
```



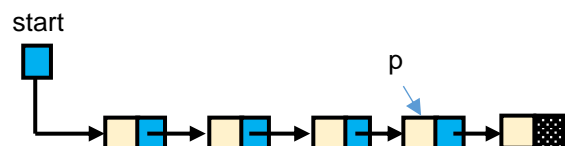
After the loop terminates, p is null

```
p = start;
while(p.link!=null)
{ .....
  p = p.link; }
```



After the loop terminates, p refers to last node

```
p = start;
while(p.link.link!=null)
{ .....
  p = p.link; }
```



After the loop terminates, p refers to second last node

When the loop condition is `while(p!=null)`, the loop executes till p is not null, it terminates when p becomes null.

So after the loop terminates, p will be null. We use this condition when we want to visit each node of the list till the last node, for example when we have to print values in all nodes or when we have to search for a value in the list. This condition we have used in the previous lecture in traversal, where we had to access info of all the nodes starting from first node till the last node.

.....

When the loop condition is `while (p.link!=null)`, the loop executes till `p.link` is not null, it terminates when `p.link` becomes null. null is present in the link of last node, so after this loop terminates p will refer to the last node.

We can use this loop when we want a reference to last node of the list. We will use it when we have to insert a new node after the last node because in that case we need a reference to the last node.

When you want to find reference to last node, you don't want to stop the loop when p becomes null, you want to stop it when p refers to the last node. So you have to write the condition `while (p.link!=null)`. If you write `while (p!=null)`, then when the loop stops, p will be null.

So write `while (p!=null)`, when you want to loop till p becomes null.
And write `while (p.link!=null)`, when you want to loop till p refers to last node.

We have used the condition `while (p.link!=null)` when finding reference to predecessor of a node with particular info.

```
p=start;
while (p.link!=null)
{
    if (p.link.info==x)
        break;
    p=p.link;
}
```

This is because in this case we want the loop to stop when p refers to the last node. We don't want the loop to execute when p refers to last node, because inside the loop we are checking `p.link.info`.

If we write `while (p!=null)`, then in the last iteration of the loop, p will refer to the last node and `p.link.info` will give error because `p.link` is null.

.....

When the loop condition is `while (p.link.link!=null)`, the loop executes till `p.link.link` is not null, it terminates when `p.link.link` becomes null and `p.link.link` will become null when `p` refers to the second last node.

So after this loop terminates `p` will refer to the second last node.

You can use this loop when you want reference to second last node of the list. We will use it when we have to delete the last node because in that case we need a reference to the second last node.

.....

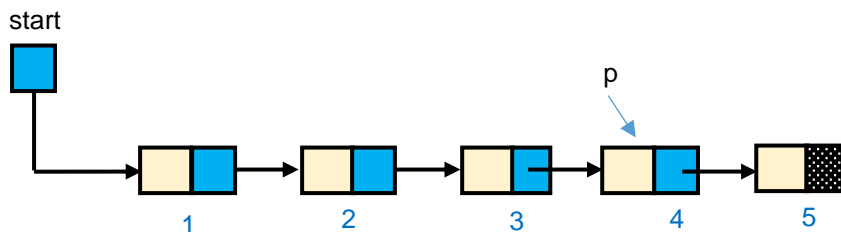
FAQ 2:

Here is a doubt that comes up in the lecture on insertion, it is better to clarify this point here only.

We have seen that this is the code for finding reference to a node at particular position.

```
p=start;
for(i=1; i<k && p!=null; i++)
    p=p.link;
```

If `k` is 4, then after the above loop terminates, `p` will refer to the fourth node.



Now suppose we need reference to the predecessor of `k`th node, so if `k` is 4, then we need reference to third node.

For this we will just make a small change in the above code.

```
p=start;
for(i=1; i<k-1 && p!=null; i++)
    p=p.link;
```

We changed k to $k-1$ and this will give us reference to predecessor of k th node. We will need this reference when we will insert a new node at the k th position in the list.

