**Why do we need to use &Head in Single Link List**

Let’s declare a pointer variable of type int  

\[
\text{int *P}
\]

So, \(P\) will point to a memory address, say 7860 and \(*P\) will give us the value stored at address 7860. Pictorially,

\[
\begin{array}{c}
\text{P} \\
\downarrow \\
\text{7860} \\
\end{array}
\]

\[
\begin{array}{c}
\text{*P} \\
\end{array}
\]

\[
\begin{array}{c}
3
\end{array}
\]

Now, suppose we have the following link list

```
ORIGINAL_HEAD 71b4
2f50
1
2
3
4
```

In this link list, \&ORIGINAL_HEAD = 71b4 and ORIGINAL_HEAD contains the address of the first node or head node of the link list, i.e. 2f50.

<table>
<thead>
<tr>
<th>*P</th>
</tr>
</thead>
<tbody>
<tr>
<td>P points to address</td>
</tr>
<tr>
<td>*P points to value</td>
</tr>
<tr>
<td>stored at that address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>**headRef</th>
</tr>
</thead>
<tbody>
<tr>
<td>headRef = 71b4</td>
</tr>
<tr>
<td>*headRef = 2f50</td>
</tr>
<tr>
<td>**headRef = 1 (value)</td>
</tr>
</tbody>
</table>

**Function call:** add_to_the_list (&ORIGINAL_HEAD)

**Function Definition:** add_to_the_list (rule_list_type **headRef)

Now, suppose we create a new node with value 0 and the address of the new node is 2f98.

\[
\text{newNode = malloc(sizeof(struct rule_list_type))) == NULL)
\]

Suppose, this new node need to be the head of the list. To change the contents of ORIGINAL_HEAD, we simply write –

\[
*\text{headRef} = \text{newNode};
\]

Now, \(*\text{headRef} = 2f98\) instead of 2f50.