

Today - Lecture 17 - CS162

1) Continue Creating Recursive Solutions!

Announcements:

* Practice!

- LLL insert, remove, traversal
- recursion

Example #3 of Topic 5


1)  Do Nothing

2) 

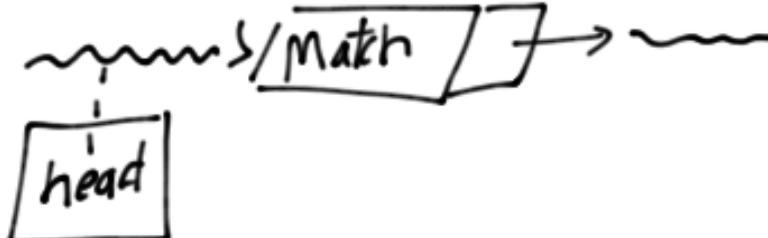
Iterative

3) 

4) No Match - Do Nothing!

1)  head \rightarrow Do Nothing

- EMPTY
- NO Match

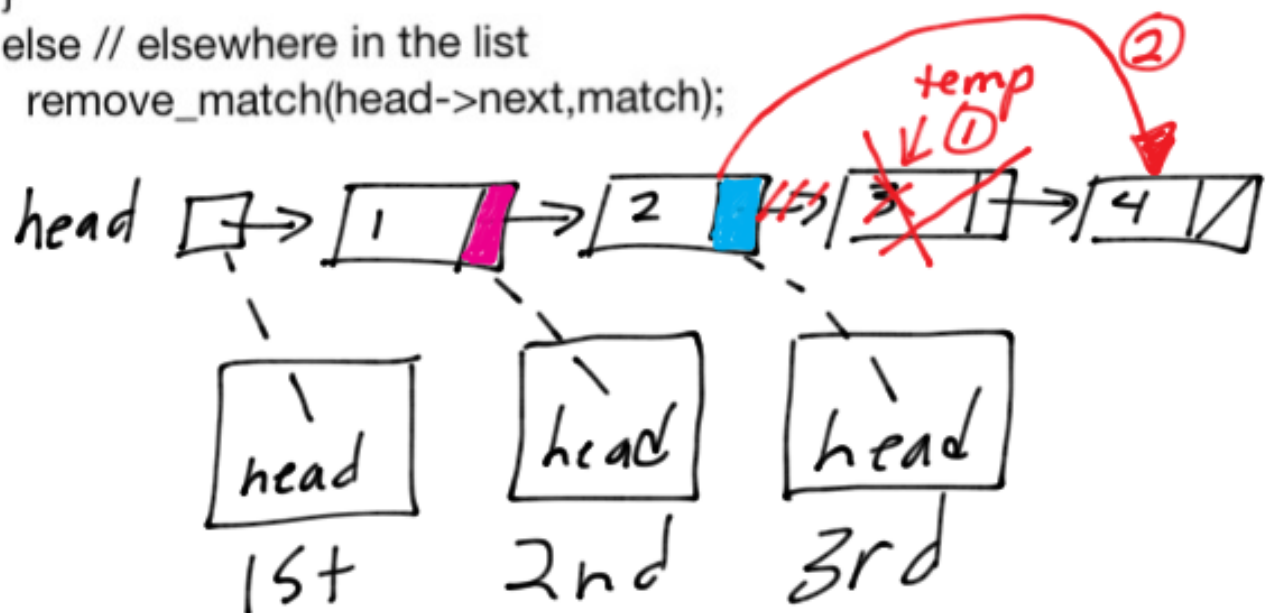
2) 

```

void remove_match(node * & head, char match[])
{
    node * temp;
    //case 1 - empty list OR no match
    if (head == NULL) //if (!head)
        return;

    //case 2 - found a match!
    if (strcmp(head->name, match) == 0)
    {
        ① temp = head; //node to remove
        ② head = head->next;
        delete [] temp->name; //dyn. mem
        delete temp; //1st node
    }
    else // elsewhere in the list
        remove_match(head->next, match);
}

```



```
void remove_match(node * & head, char match[])
```

```
{
```

```
    node * temp;
```

```
    //case 1 - empty list OR no match
```

```
    if (head == NULL) //if (!head)
```

```
        return;
```

```
    //case 2 - found a match!
```

```
    if (strcmp(head->name, match) == 0)
```

```
    {
```

```
        temp = head; //node to remove
```

```
        head = head->next;
```

```
        delete [] temp->name; //dyn. mem
```

```
        delete temp; //1st node
```

```
    }
```

```
    else // elsewhere in the list
```

```
        remove_match(head->next, match);
```

```
}
```

what if we forgot this?



Alternative — Viablr

*node ** remove_match(*node ** head, char match[]) *by value*

```
{
    node * temp;
    //case 1 - empty list OR no match
    if (head == NULL) //if (!head)
        return NULL;

    //case 2 - found a match!
    if (strcmp(head->name, match) == 0)
    {
        temp = head; //node to remove
        head = head->next;
        delete [] temp->name; //dyn. mem
        delete temp; //1st node
    }
    else // elsewhere in the list
        head->next = remove_match(head->next, match);
} return head;
```

*head = remove_match(
head, "3");*

*First time the
function is
called.*

head →

1

 →

2

 →

3

 →

4



Alternative: Passing by Pointer (simulates pass by reference)

```
void remove_match(node** head, char match[])  
{  
    node * temp;  
    //case 1 - empty list OR no match  
    if(*head == NULL) //if (!head)  
        return;
```

"head" is a pointer to a pointer to a node

```
    //case 2 - found a match!  
    if (strcmp((*head) -> name, match) == 0)  
    {  
        temp = *head; //node to remove  
        head = (*head) -> next;  
        delete [] temp->name; //dyn. mem  
        delete temp; //1st node  
    }  
    else // elsewhere in the list  
        remove_match(head -> next, match);  
}
```

&(*head) -> next

Address-of operator

remove_match(&head, "3");

this is how you call the function the first time