

Data Structures: Workshop 4

Q1. Convert the following binary number to hexadecimal

1100 0101 1111 0001

12 5 15 1

C 5 F 1

decimal	hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Q2

0001 1111 0111 1010

1 F 7 A

Q3

```
node* deleteCell(node* header, int val)
{
    boolean found = false;
    node *current = header;
    node*previous = NULL;

    while (!found && current!= NULL)
    {
        if (current->data == val){
            found = true;
        }
        else{
            previous = current;
            current = current->next;
        }
    }

    previous->next = current->next;
    current = NULL;
    return header;
}
```

1. Empty Chain
header = NULL;
current = NULL;

Runtime error: as current is NULL



Q3

```
node* deleteCell(node* header, int val)
{
```

```
    boolean found = false;
    node *current = header;
    node*previous = NULL;
```

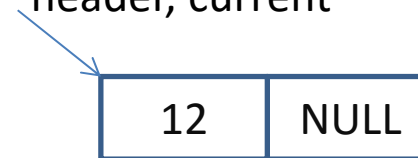
2. List has only one node, value present

```
    while (!found && current!= NULL)
    {
```

```
        if (current->data == val){
            found = true; ← previous is not set
```

val = 12;
header, current

```
        }
        else{
            previous = current;
            current = current->next;
        }
    }
}
```



```
    previous->next = current->next;
    current = NULL;
    return header;
```

Runtime error: as previous is NULL

```
}
```

Q3

```
node* deleteCell(node* header, int val)
{
    boolean found = false;
    node *current = header;
    node *previous = NULL;

    while (!found && current != NULL)
    {
        if (current->data == val){
            found = true;
        }
        else{
            previous = current;
            current = current->next;
        }
    }

    previous->next = current->next;
    current = NULL;
    return header;
}
```

3. Value not present

Runtime error: as current is set to NULL at the end of the while loop

previous->next = current->next; ←

4. Work through an example for each boundary condition

- **empty chain**
(fails at previous -> next = current -> next)
- **List has only one node, value present**
(fails at previous -> next = current -> next)
- **value not present**
(fails at previous -> next = current -> next)

```
node* deleteCell(node* header, int val)
{
```

```
    boolean found = false;
    node *current = header;
    node *previous = NULL;
```

```
    if (header == NULL) return NULL;
```

1. Empty chain

```
    if (header->next == NULL && header->data == val)) return NULL;
```

2. List has one node, value presents

```
    // general case
    while (!found && current!=null) {
        if (current->data == val)
            found = true;
        else{
            previous = current;
            current = current->next;
        }
    }
```

```
    if (found){
        if (current == header){
            return header -> next;
        }
        else{
            previous->next = current->next;
            current = NULL;
        }
    }
```

3. Only remove a node if value presents

```
    return header;
```

```
}
```