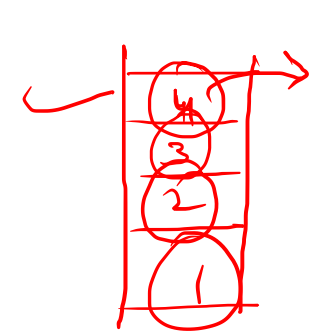
Delete Middle Element of a Stack

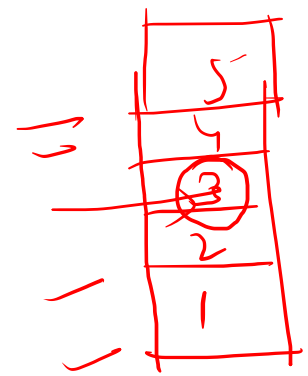
LIFO



I/p: st = < 1, 2, 3, 4 >

O/p: < 1, 3, 4 >

Ceil((n+1)/2)



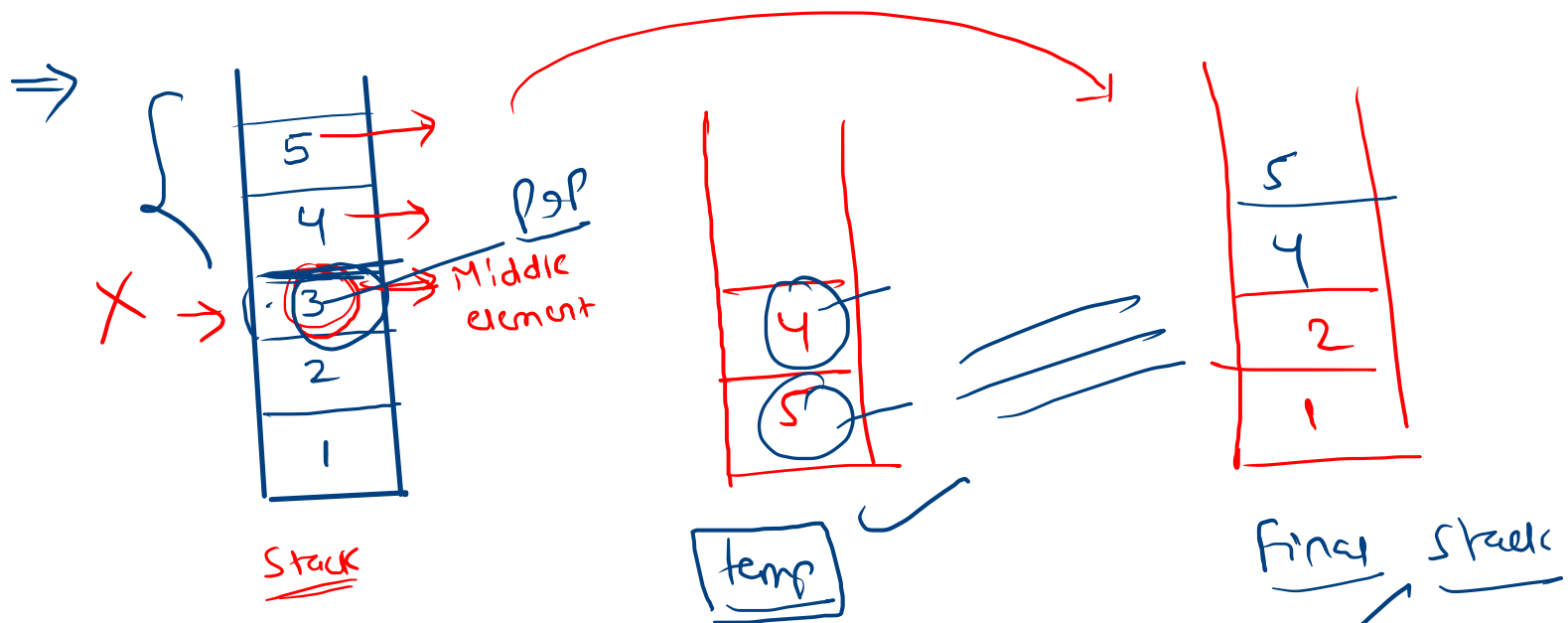
I/p: st = < 1, 2, 3, 4, 5 >

O/p: < 1, 2, 4, 5 >

NAIVE SOLUTION

I/p: st = < 1, 2, 3, 4, 5 >

O/p: < 1, 2, 4, 5 >

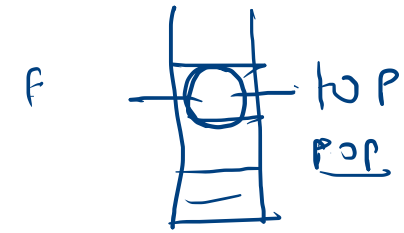


O(n/2)

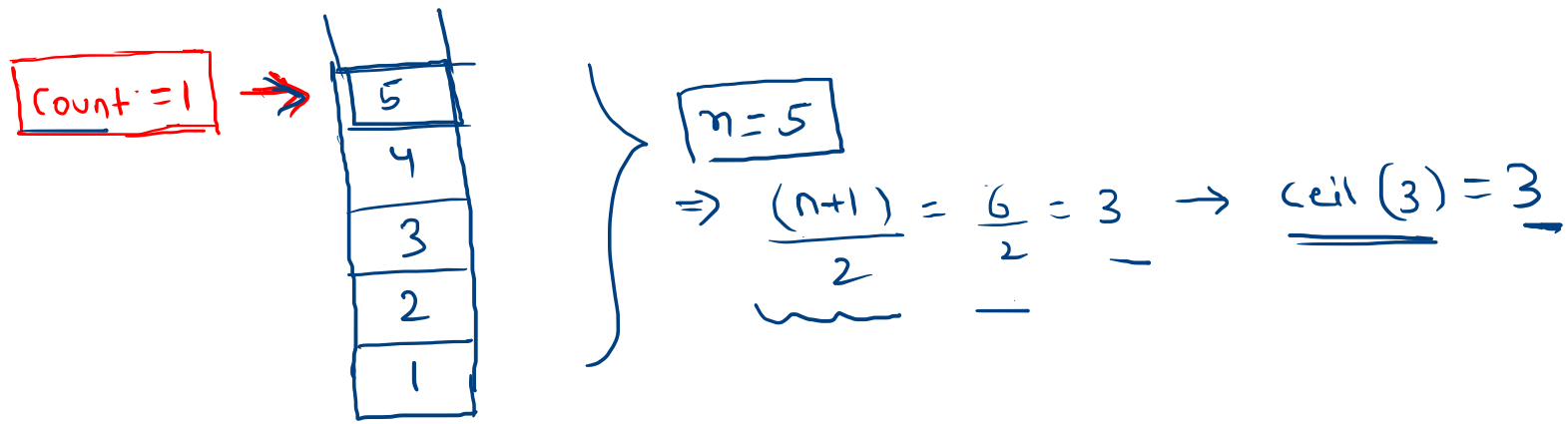
Recursive Solution

I/p : st = < 1, 2, 3, 4, 5 >

o/p : < 1, 2, 4, 5 >



Logic



Recursive Solution

I/p : st = < 1, 2, 3, 4, 5 >

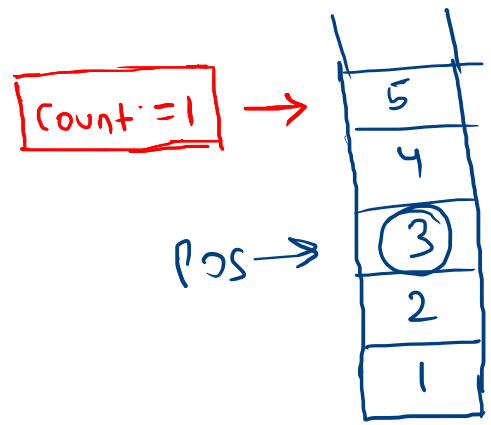
o/p : < 1, 2, 4, 5 >

5/2 → 2 → ceil → 2

5/2.0 → 2.5 → ceil → 3

ceil((n+1)/2) → pos

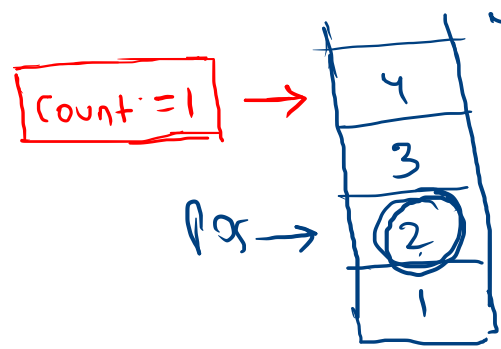
Logic



pos = 3

n = 5

⇒ (n+1)/2 = 6/2 = 3 → ceil(3) = 3



n = 4

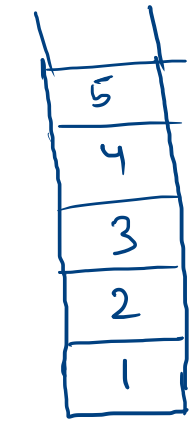
⇒ (n+1)/(2.0) = 5/2.0 = 2.5 → ceil(2.5) = 3

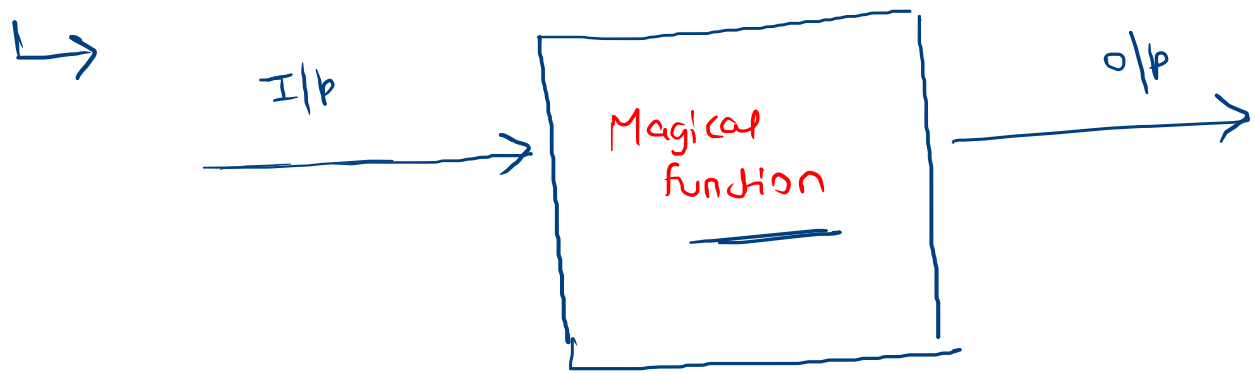
pos

Recursive Solution

I/p : st = < 1, 2, 3, 4, 5 >

o/p : < 1, 2, 4, 5 >

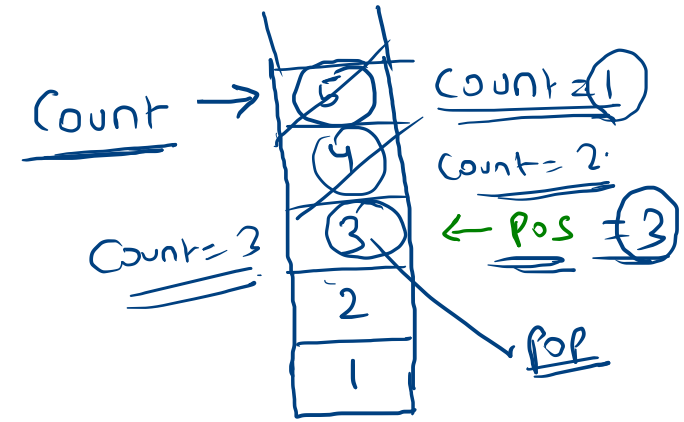


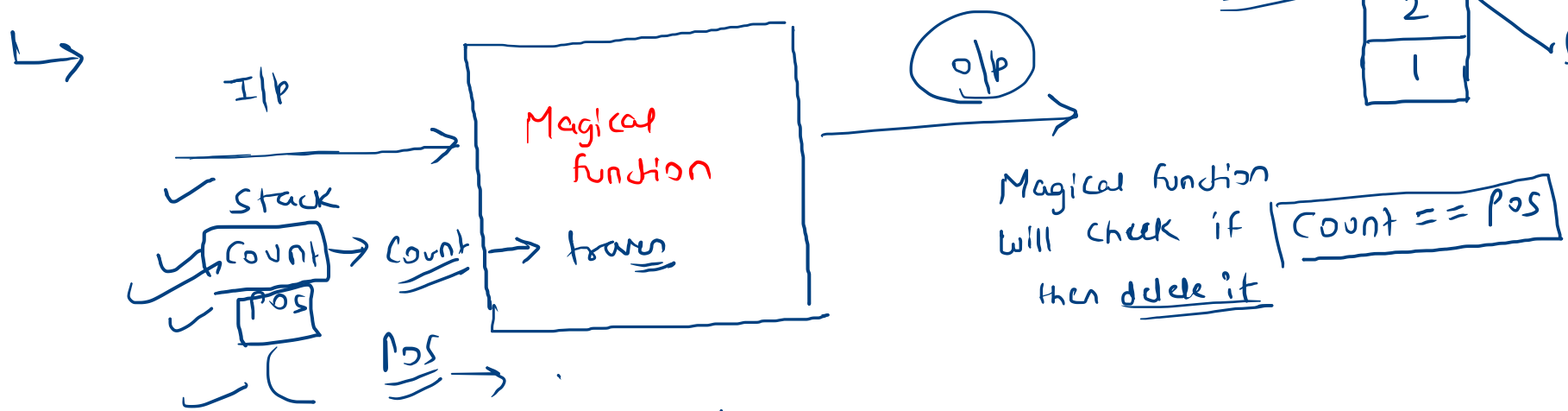


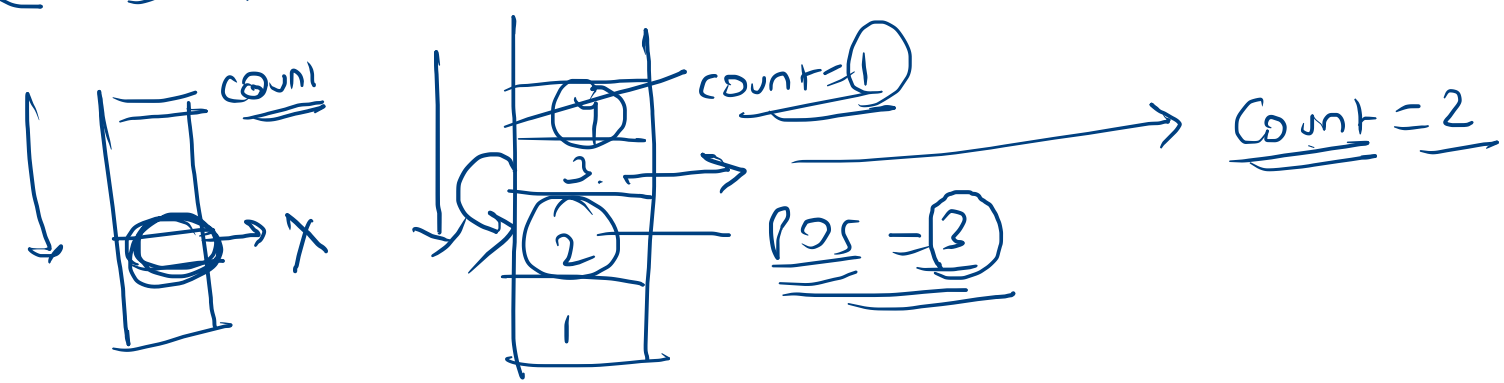
Recursive Solution

I/p : st = < 1, 2, 3, 4, 5 >

o/p : < 1, 2, 4, 5 >



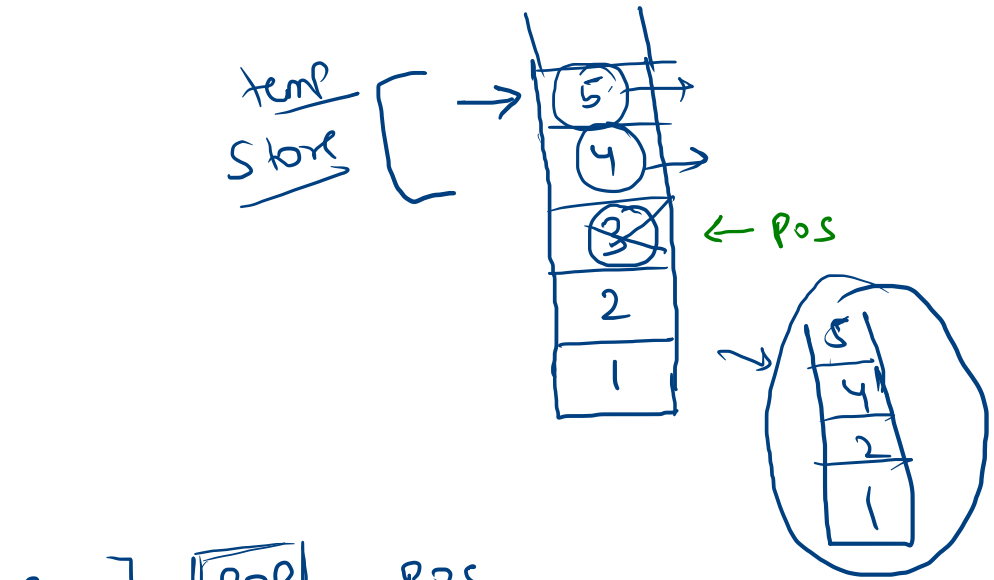




Logical Part

I/p : st = < 1, 2, 3, 4, 5 >

o/p : < 1, 2, 4, 5 >



pos = 3

↳ st = < 1, 2, 3, 4, ~~5~~ >

Count Count = 1

Function checks if [Count == Pos] Pop .Pos

if not

then, Pop Count and store element in temp Variable

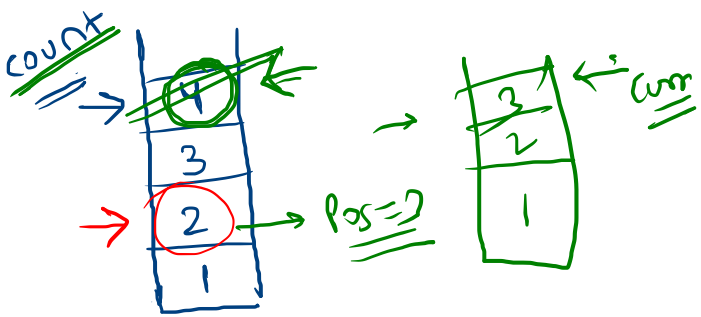
and Call Recursive function for Count + 1

Recursive code

```
Void delete ( stack <int> st, int count, int pos)
{
    // Base condition

    int top = st.top();
    st.pop();
    delete (st, count+1, pos)

    st.push(top);
}
```



Base condition

Smallest Valid Input → Check Output

Void delete (stack <int> st, int count, int pos)

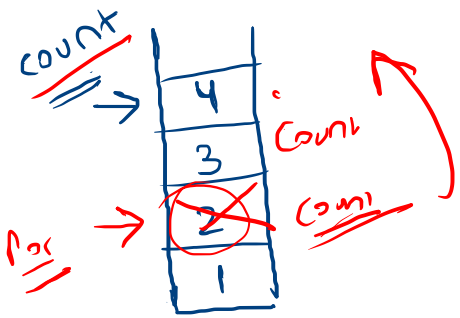
↓

1. if stack is empty

```
if ( st.empty () ) return ;
```

Base condition

Smallest Valid Input → Check output



Void delete (stack <int> st, int count, int pos)

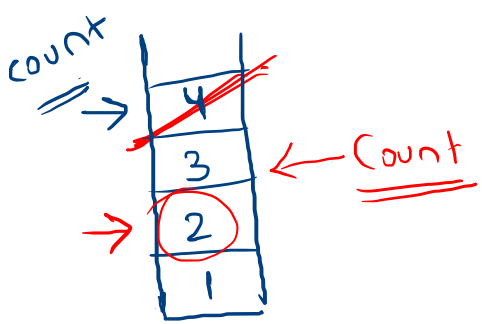
① if stack is empty

```
if ( st.empty () ) return
```

② (if count == pos)

```
{
    st.pop ();
    return;
}
```

Recursive code




```
Void delete ( stack <int> st, int count, int Pos)
{
   [ if( st.empty() ) return;
     if( count == Pos ) { st.pop(); return; }        ] Base condition

   {  int top = st.top();
      st.pop();
      delete (st, count+1, Pos)
      st.push(top);                                  } Logics
}
```

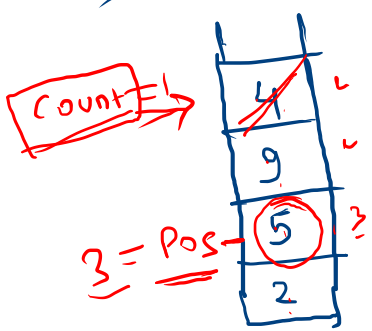
Dry Run

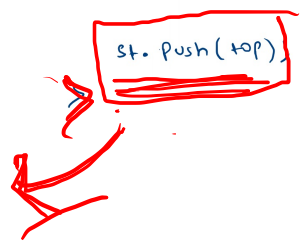
(1)

Count = 1 , Pos = 3

```
Void delete ( stack <int> st, int count, int Pos)
{
    if( st.empty() ) return;
    if( count == Pos ) { st.pop(); return; }      ] Base condition

    int top = st.top(); // 4
    st.pop();
    delete (st, count+1, Pos)                     ] Logics
    st.push(top);
}
```

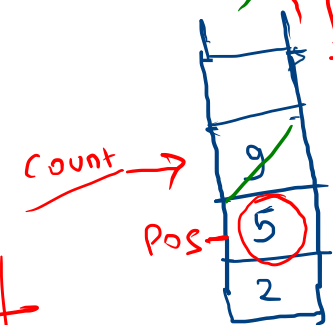




Count = 2 Pos = 3

```
Void delete ( stack <int> st, int count, int Pos)
{
    if( st.empty() ) return;
    if( count == Pos ) { st.pop(); return; }      ] Base condition

    int top = st.top(); // 9
    st.pop();
    delete (st, count+1, Pos)                     ] Logics
    st.push(top);
}
```



Count = 3 Pos = 3

```
Void delete ( stack <int> st, int count, int Pos)
{
    if( st.empty() ) return;
    if( count == Pos ) { st.pop(); return; }      ] Base condition

    int top = st.top();
    st.pop();
    delete (st, count+1, Pos)                     ] Logics
    st.push(top);
}
```