Reverse a String

Ilp: str = "Geeks"

Olp: "skeeG"

Ilp: str = "a"

Olp: "a"

Ilp: str = "abc"

Olp: "cba"

Ilp: str = "" (Empty string)

Olp: ""

NAIVE APPROACH

Ip: str = "Geeks" (0,1,2,3,4)

Op: "skeeG"

n = 5

Code:

```
for(int i=n-1; i>=0; i--)
{
    ans += str[i];
}
cout << ans << endl;
```

```
int n = str.size();
string ans = "";
```

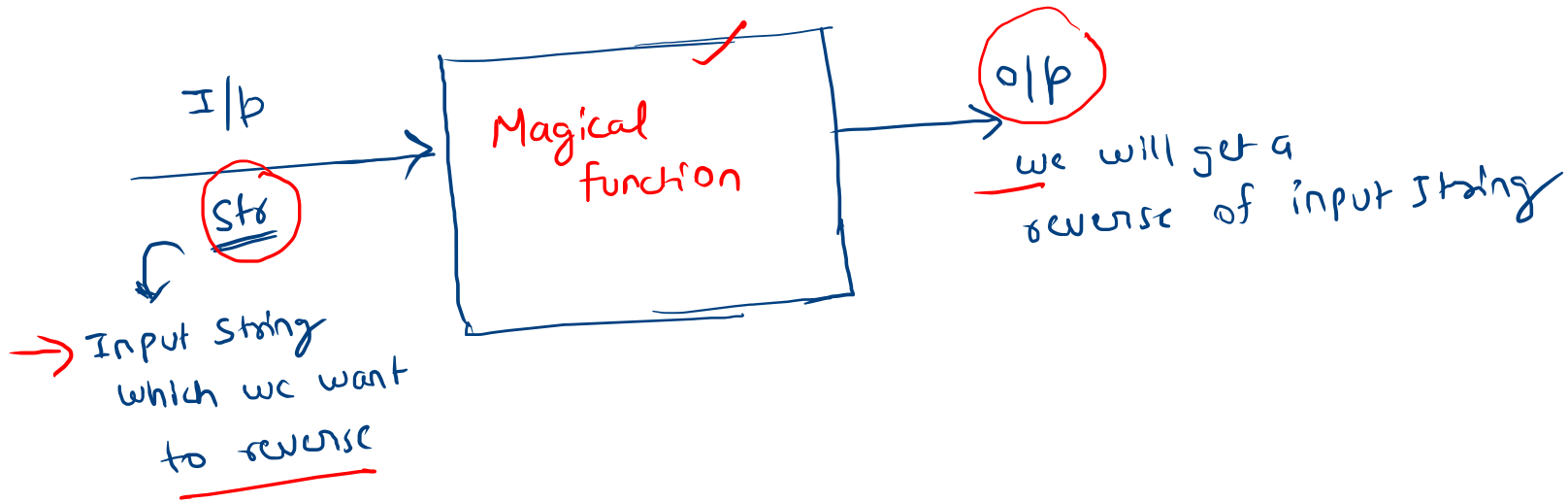
skeeG

Geeks

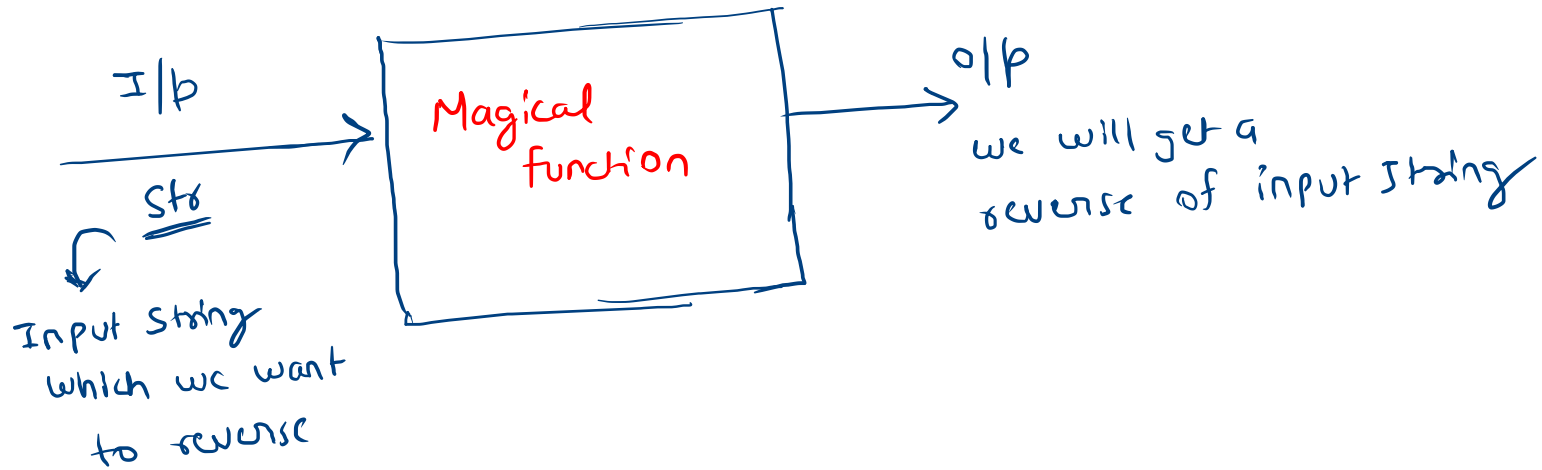
Recursive Solution

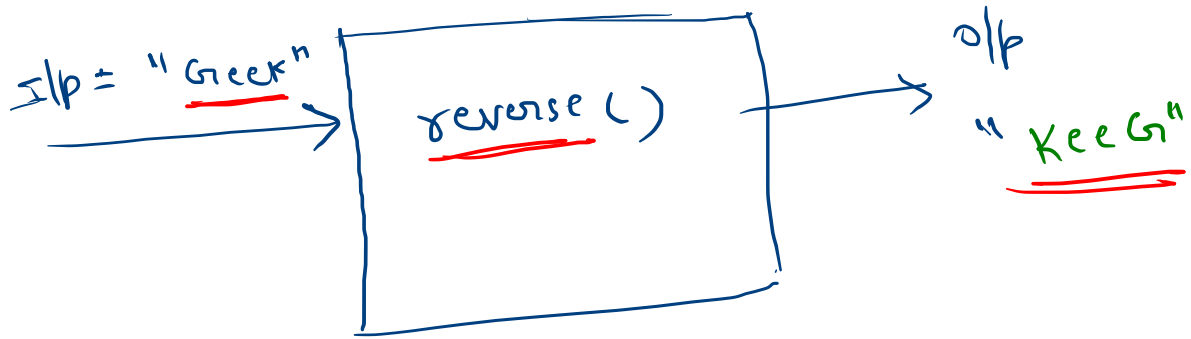
I/p: str = "Greeks"

o/p: "skeeG"



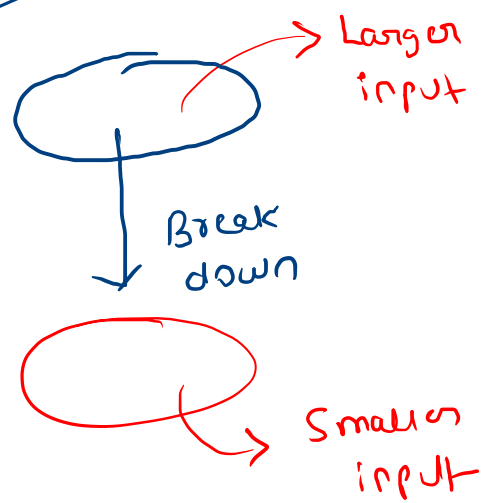
Recursive Solution





Recursive Solution

Logical Part



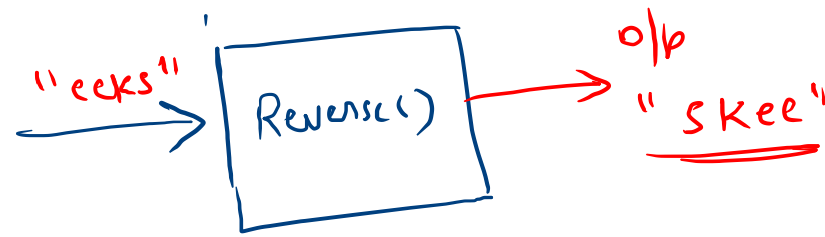
I/p: "Geeks"

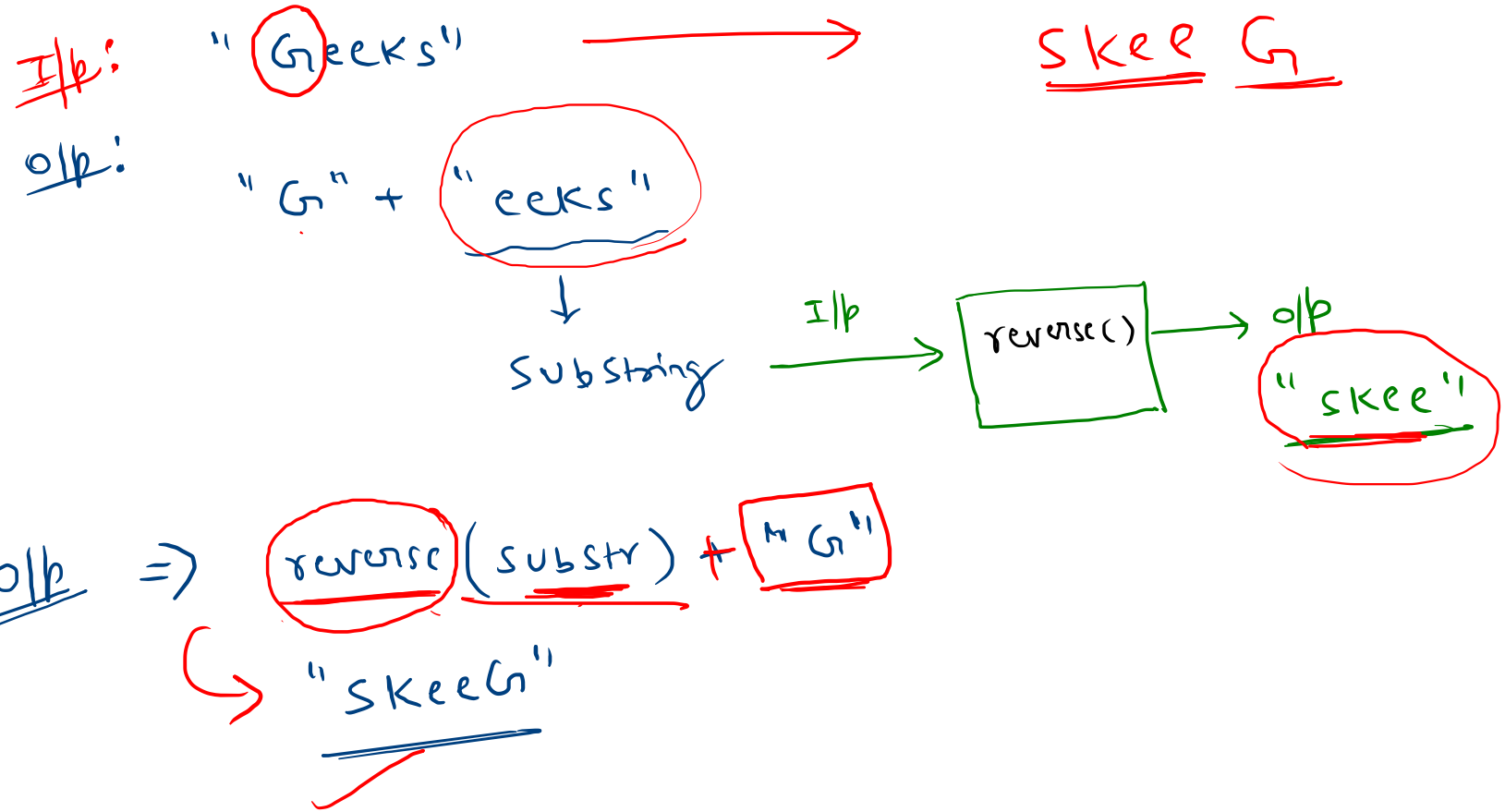
o/p: "G" + "eeks"

Substring

↓

our smaller I/p in the recursive function




I/p: "Geeks"
skee G
o/p:
"G" + "eeks"
substring
I/p
reverse()
o/p
"skee"
o/p => reverse(substr) + "G"
"skeeG"

∴ Find Substring in a given String in C++

Ilb str = " Geeks "

0 1 2 3 4

↳ in C++, there is a inbuilt function → substr()

↓

Takes 2 Parameter

(position, length)

Ex. str.substr(2,3);

olb → "eks"

str.substr(1,2);

olb → "ee"

str.substr(1);

↳ "eeks"

str.substr(3);

↳ olb: "ks"

0 1 2 3 4

G e e k s

Recursive Solution

```
String reverse (string str)
{
    // Base Condition
    return reverse (str.substr(1)) + str[0];
}
```

Recursive Solution

```
String reverse (string str)
{
    // Base Condition
    return reverse ( str.subStr(1) ) + str[0] ;
}
```

Ex → str = "Prince"

0 1 2 3 4 5

↳ "rince"

reverse ("rince")

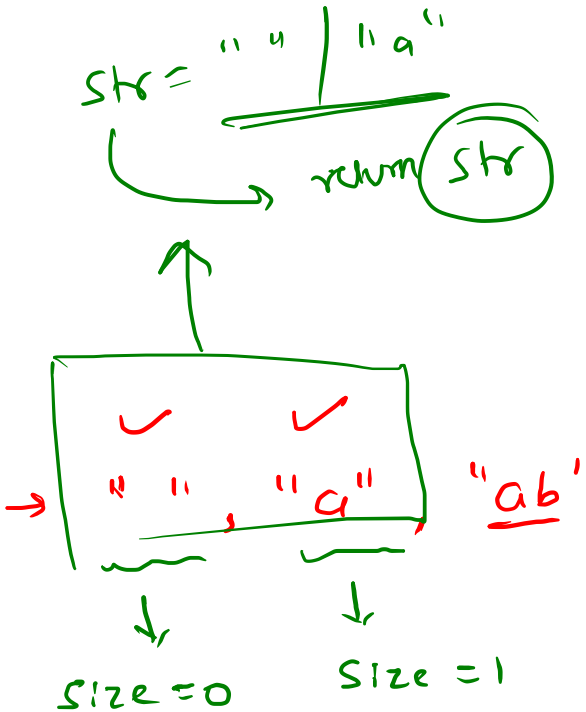
↳ ecnir

"ecnirP"

Base Condition : Smallest Valid Input → check output

String reverse (string Str)
{
}

Input
↳ Example of input → "", "a", "ab"

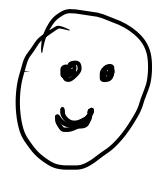



```
if( str.size() == 0 || str.size() == 1 ) return str;
```

Recursive Solution

```
String reverse (string str)
{
    if(str.size()==0 || str.size()==1) return str;
    return reverse(str.substr(1)) + str[0];
}
```

Home work → Dry Run and function call for I/P : "abc"



Please