

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

Leetcode May Challenge DAY: 13

1. Python

class Solution:

def removeKdigits(self, num: str, k: int) -> str:

stack = []

for x in num:

while stack and stack[-1] > x and k: # stack[-1]check to verify there are elements in stack

stack.pop()

k -= 1

stack.append(x)

return "".join(stack[:len(stack)-k]).lstrip("0") or "0"

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

2. C++

```
class Solution {  
public:  
    string removeKdigits(string num, int k) {  
        for (int i=0; i<num.length()&& k>0; i++){  
            if (i==num.length()-1 || num[i]>num[i+1]){  
                num.erase(i, 1);  
                i-=2; k--;  
                if (i<-1) i=-1;  
            }  
        }  
        int ind = 0;  
        while (ind<num.length() && num[ind]=='0'){  
            ind++;  
        }  
        return (ind==num.length()?"0":num.substr(ind));  
    }  
};
```

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

3. JAVA

```
class Solution {  
    public String removeKdigits(String num, int k) {  
        int len = num.length();  
        if (len == k){  
            return "0";  
        }  
  
        //whenever we find decreasing, we replace the original one with current one  
        //eg. 1432219  
        //14 -> 13 -> 12 -> 12 -> 11  
        //only deleting 1 digit, we can achieve the smallest  
  
        int counter = 0;  
        Stack<Character> stk = new Stack<>();  
        for (; counter < len; counter++) {  
            char curDigit = num.charAt(counter);  
            while (!stk.isEmpty() && stk.peek() > curDigit && k > 0) {  
                //replace all previous larger digit with cur digit(if k permits)  
                stk.pop();  
                --k;  
            }  
            stk.push(curDigit);  
        }  
  
        //corner case: "1111", cannot find decreasing -> no deletion  
        while (k > 0) {
```

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

```
        stk.pop();  
        --k;  
    }  
  
    StringBuilder sb = new StringBuilder();  
    while (!stk.isEmpty()) {  
        sb.append(stk.pop());  
    }  
    sb.reverse();  
  
    //remove 0 at beginning, except 0  
    while (sb.length() > 1 && sb.charAt(0) == '0') {  
        sb.deleteCharAt(0);  
    }  
  
    return sb.toString();  
}  
}
```