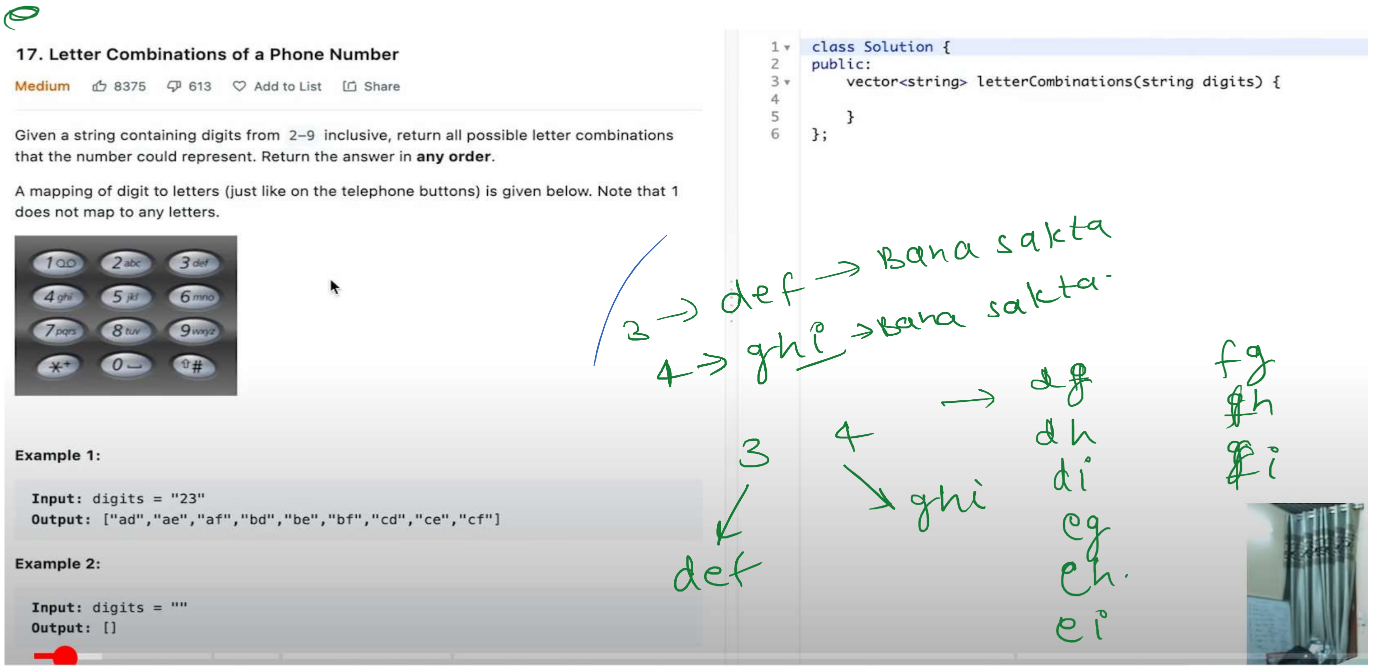
Code 8

27 May 2022 16:40



17. Letter Combinations of a Phone Number

Medium 8375 613 Add to List Share

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

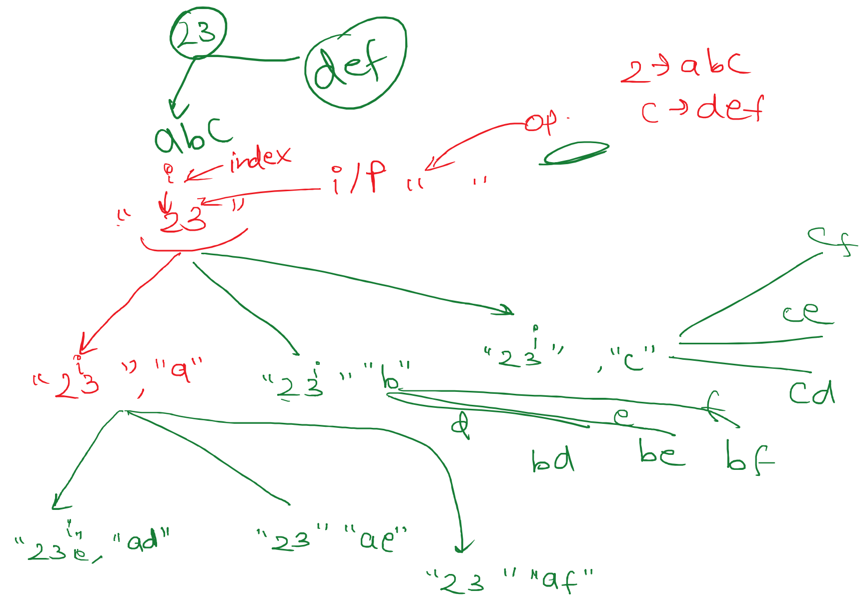
```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

Example 2:

```
Input: digits = ""
Output: []
```

```
class Solution {
public:
    vector<string> letterCombinations(string digits) {

    }
};
```



9 Answer getting :-

```
vector <string> lettercombination          234
        (string digits) {
    vector<string> ans;          o/p= " "
    int index = 0;  → string      index
                      i/p          mapping
    string mapping [10]
{if (digits.length()==0)    [ 0 = " " "   } re  sab
```

{if (digit.length()==0)
return ans;
}

string mapping[10]

0 = " "
1 = " "
2 = "abc"
9 = "wxyz"

re sab pada hai

solve(digit, output, index,
ans, mapping)

} return ans;

```
void solve (string digit, string o/p, int index
    vector< string> &ans, string mapping[])
{
    //base case · jab index digit se bahar
    if( index >=digit.length())
    { ans.push_back (output);
        return;
    }
```

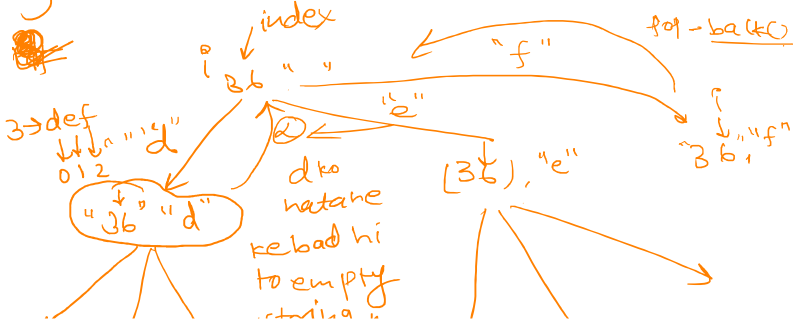
jis bhi index ko no. point kar raha hai uske saath kuch kuch karna hai

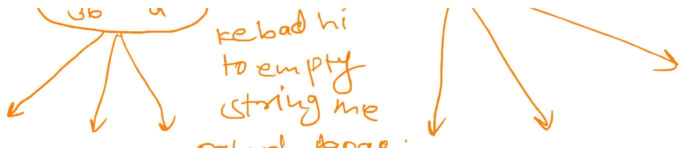
exact no.

```
    int number = digit[index] - '0';
                 char
    string value = mapping [number];   → iski string value nikal &
    for(int i=0 ; i<value.length(); i++)   → ek bar a, b, c
    { output.push_back(value[i]);
      solve (digit, output, index+1, ans,
             mapping)
```

Ab ya se a hatana bhi padega.

```
    output.pop_back();   // backtracking
    }
}
```





Dynamic Memory Allocation :-

$\boxed{\text{int i = 5;}}$ → static Allocation.

$\boxed{\text{int *ptr = new int}}$

$\underbrace{\text{new int}}_{\text{address}}$

↑ dynamic Allocation

Recursion :- When a f^n calls itself :-

mandatory

→ $\boxed{\text{Base case}}$ → Rukna kaha hai → terminating condition.

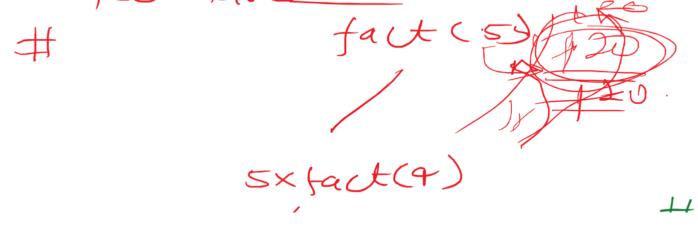
→ Recursive call / Recursive Relation.

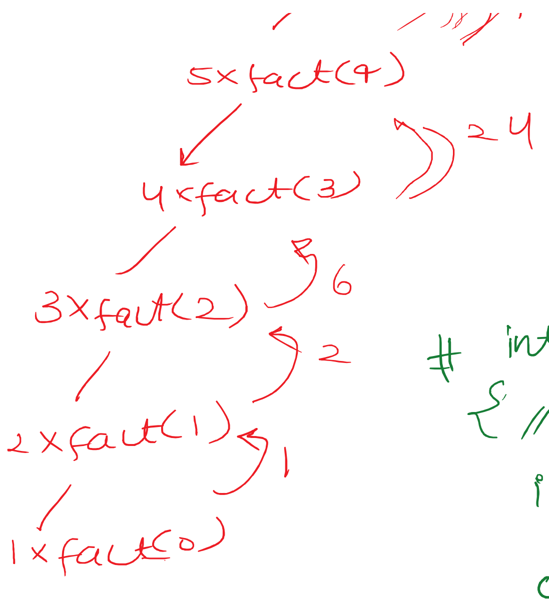
~~factorial~~

5! = ~~4! ×~~ 5×4!
4! = 4×3!
3! = 3×2!
2! = 2×1!
1! = 1×0!

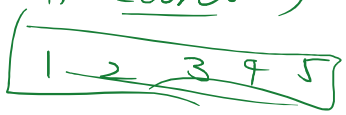
```
if(n==1 || n==0)
    return 1.
int smallerpart = fact(n-1)
int biggerproblem = n*smalle
return biggerproblem;
```

Recursive Tree :-



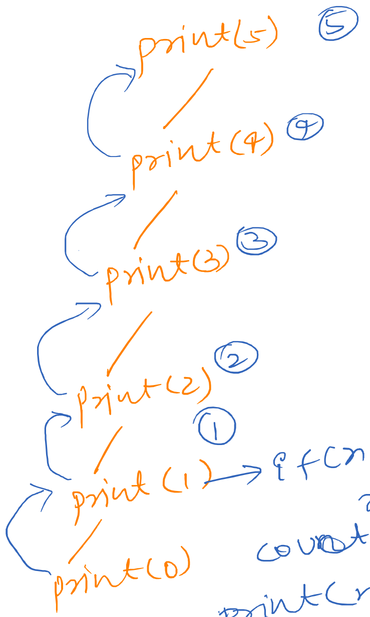


Counting



if

```
# int count(int n)
{ //base case:-
    if(n<1)
        return
    count(n-1)
} cout<<n<<" ";
```



```
→ if(n==1)
        return;
  count<<n;
  print(n-1);
```

Que:- fastexponential:-

lets say:- $2^n \rightarrow 2^{n/2} * 2^{n/2}$ (is n is even)

if n is odd then $2^n \rightarrow 2 * 2^{n/2} * 2^{n/2}$.

Base case:-
```
int exp(int n)
    if(n==0)
        return 1;
    int chhota problem = exp(n/2)
```

```
if(n&1)
  {   2* cp*cp;
  }
else { cp * cp;
```

questions:- coin change problem:-

D5 Sudoku Solver

8 A sudoku solution must satisfy all of the following rules:

1 Each of the digits 1-9 must occur exactly once in each row.
2 Each of the digits 1-9 must occur exactly once in each column.
3 Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

```
{ if (possible)
  {    call Recursive;
```