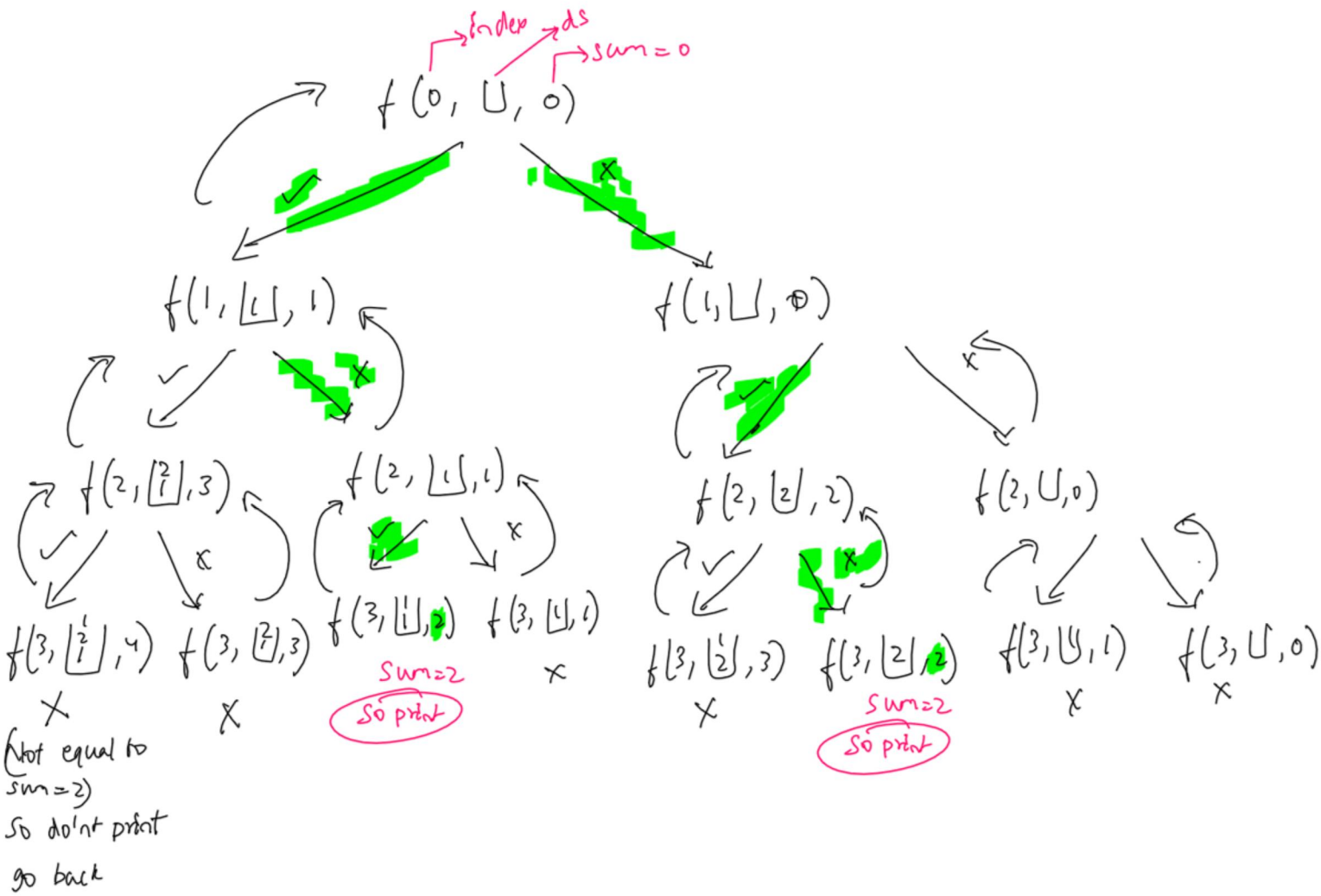
Q1: arr → [1, 2, 1] sum = 2

o/p ⇒ [1, 1]
[2]

0 1 2
arr → [1, 2, 1]



[✓ x ✓] ⇒ [1 1]
[x ✓ x] ⇒ [2]

Output
1, 1
2

Pseudo Code

f(i, ds, s)

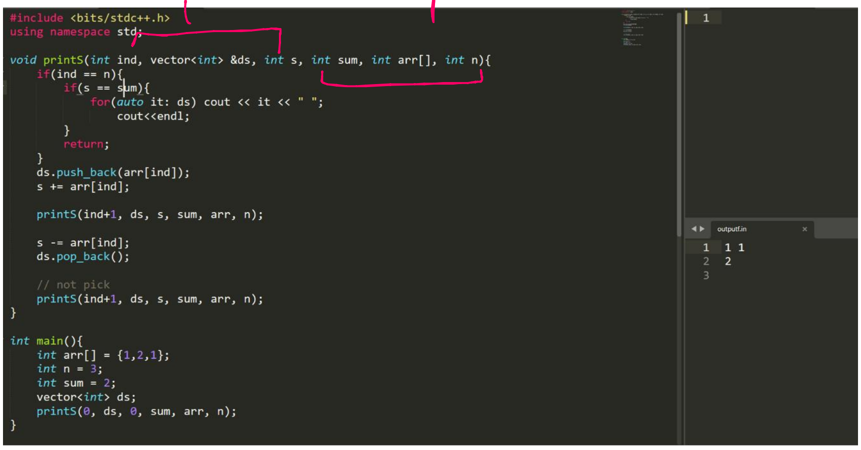
```
f(i, ds, s)
{
    if (i == n)
    {
        if (s == sum)
            print (ds)
        return;
    }
    ds.add (arr[i]);
    s += arr[i]
    f (i+1, ds, s)
    ds.remove (arr[i]); // remove array index // remove
    s -= arr[i] // need to remove array i as well from the sum // remove
    f (i+1, ds, s);
}
```

→ pick (ds.add … s -= arr[i])

→ NOT PICK (f(i+1, ds, s);)

Recursion Parameter

Given Parameter



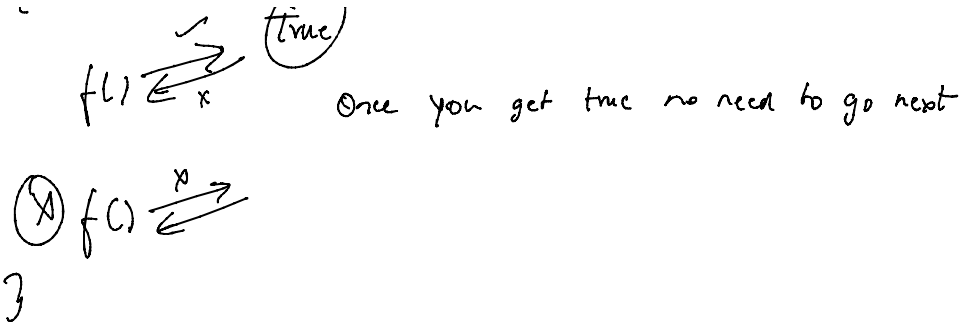
```cpp
#include <bits/stdc++.h>
using namespace std;

void printS(int ind, vector<int> &ds, int s, int sum, int arr[], int n){
    if(ind == n){
        if(s == sum){
            for(auto it: ds) cout << it << " ";
                cout<<endl;
        }
        return;
    }
    ds.push_back(arr[ind]);
    s += arr[ind];

    printS(ind+1, ds, s, sum, arr, n);

    s -= arr[ind];
    ds.pop_back();

    // not pick
    printS(ind+1, ds, s, sum, arr, n);
}

int main(){
    int arr[] = {1,2,1};
    int n = 3;
    int sum = 2;
    vector<int> ds;
    printS(0, ds, 0, sum, arr, n);
}
```

```
1 1
2
```

Q2: Print any one Subsequences whose sum is given.

```
f()
{
    f() → true
```

Once you got true no need to go next



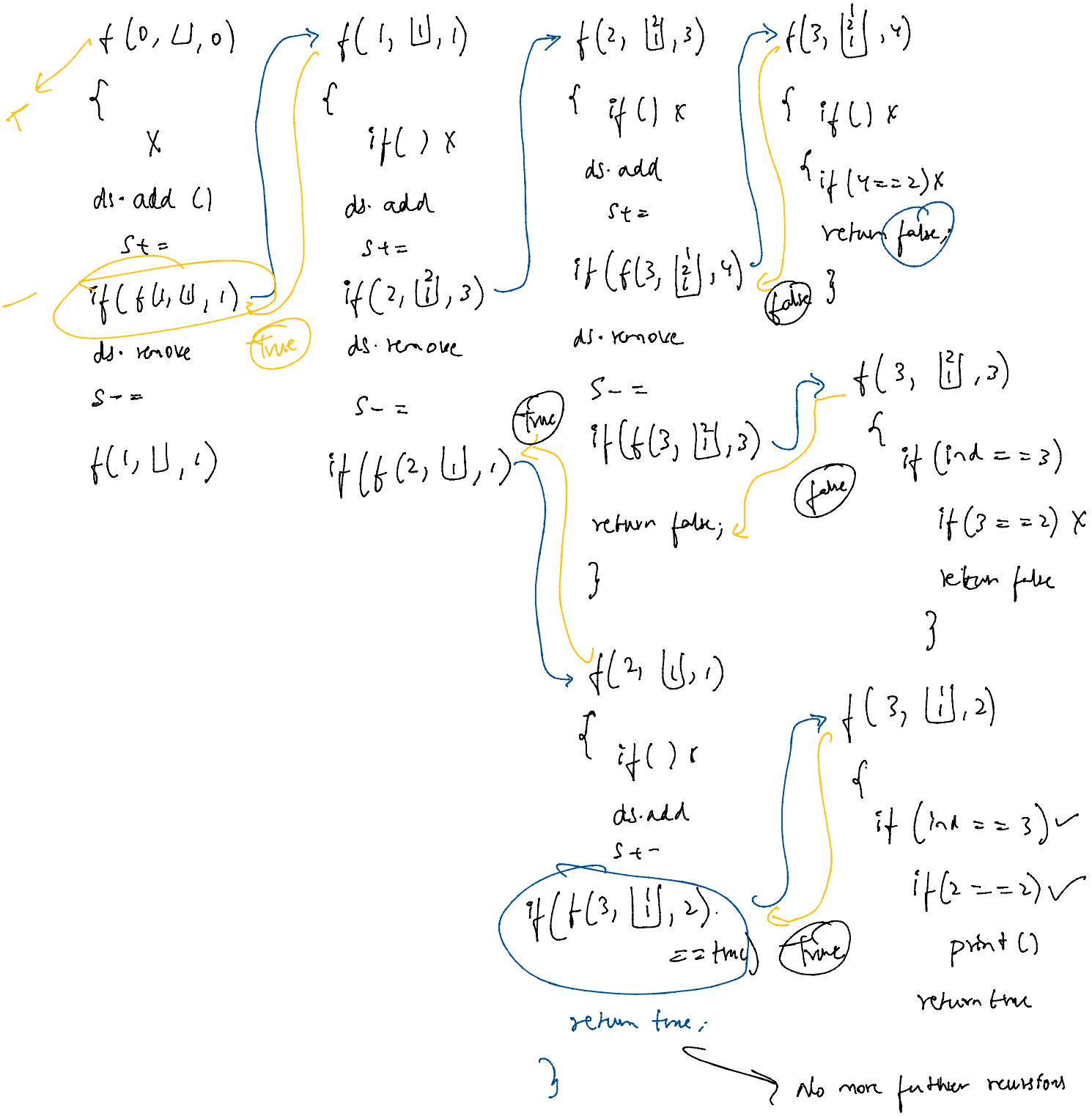
The technique to print one answer

```
bool f()
{
    base case
    Condition → Satisfied
    return (true)
        → not satisfied
    return false
    if(f() == true)
        return true;
    f()
    return false;
}
```

```cpp
bool printS(int ind, vector<int> &ds, int s, int sum, int arr[], int n) {
    if(ind == n) {
        // condition satisfied
        if(s == sum) {
            for(auto it : ds) cout << it << " ";
            cout << endl;
            return true;
        }
        // condition not satisfied
        else return false;
    }

    ds.push_back(arr[ind]);
    s += arr[ind];

    if(printS(ind+1, ds, s, sum, arr, n) == true) {
        return true;
    }

    s-= arr[ind];
    ds.pop_back();

    // not pick
    if(printS(ind+1,ds, s, sum, arr, n) == true) return true;

    return false;
}
```

o/p ⇒ 1 1

sum = 2 [1, 2, 1]



Q3 :

Count the subsequences whose sum = k

[1, 2, 1] k = 2 (2 subsequences)

```
f()
{
  base case

  return 1 ⟶ condition satisfies

  return 0 ⟶ condition not satisfies

  l = f()
  r = f()      } Here '2' recursion call

  return l+r;
}
```

for multiple recursion calls

```
S = 0
fn(i = 1 ⟶ n)
    S += f()
return S
```

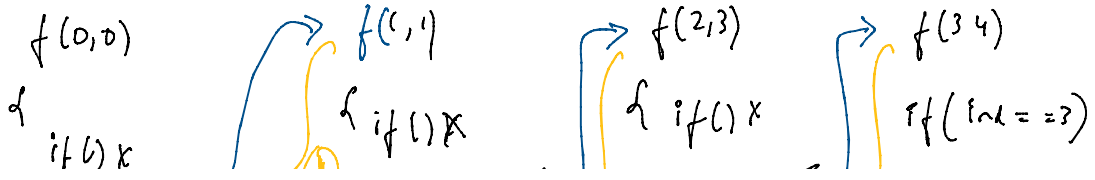
} Ex N Queen

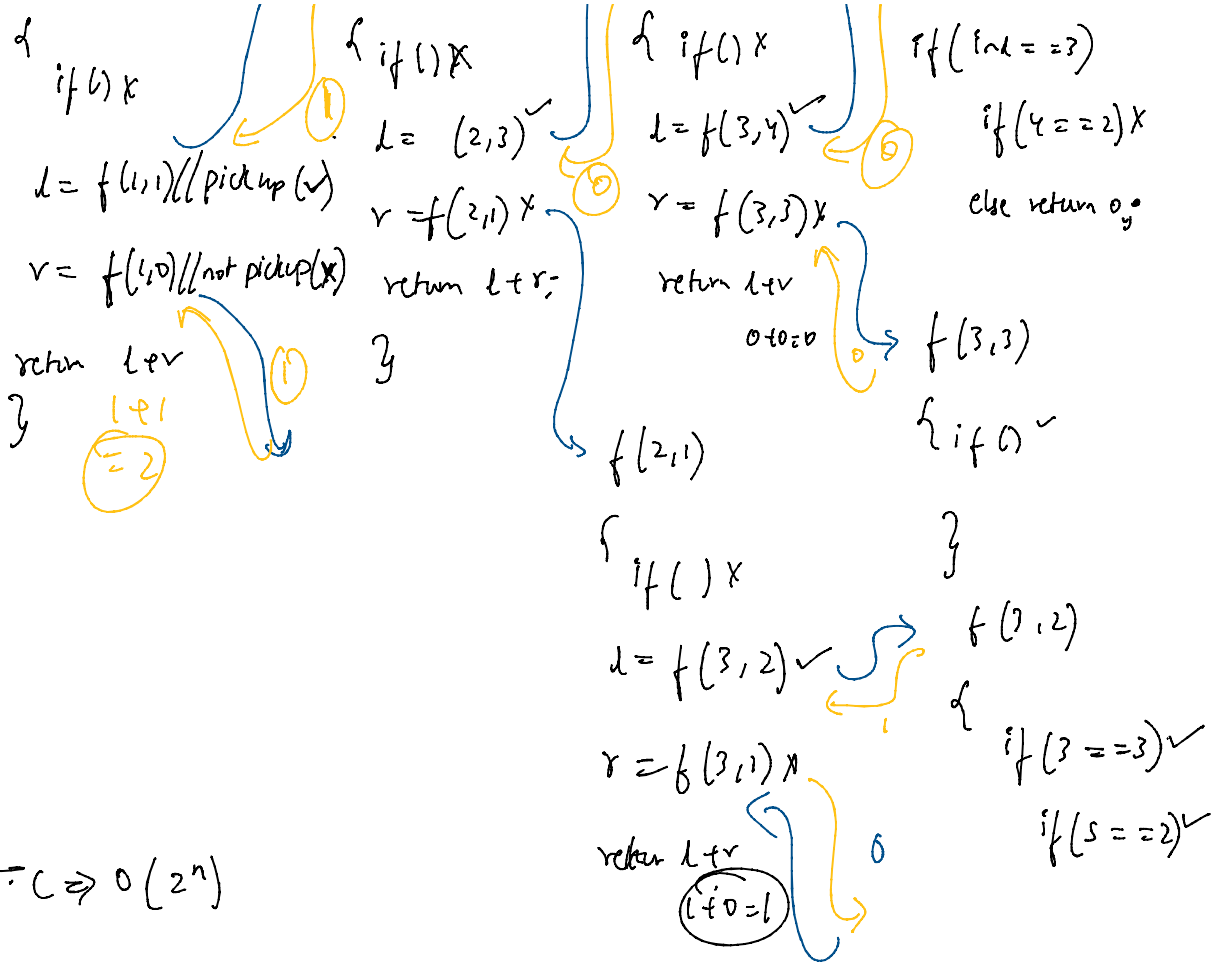
```
using namespace std;

int printS(int ind, int s, int sum, int arr[], int n) {
    if(ind == n) {
        // condition satisfied
        if(s == sum) return 1;
        // condition not satisfied
        else return 0;
    }

    s += arr[ind];

    int l = printS(ind+1,  s, sum, arr, n) ;

    s-= arr[ind];

    // not pick
    int r = printS(ind+1, s, sum, arr, n);

    return l + r;
}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {1, 2, 1};
    int n = 3;
    int sum = 2;
    cout << printS(0, 0, sum, arr, n);

    return 0;
}
```

input.txt
```
5
1 2 3 4 5
```

output.txt
```
2
```

-X-





T.C ⇒ $O(2^n)$

S.C ⇒ $O(N)$

```
    // condition not satisfied
    // strictly done if array contains positives only
    if(s > sum) return 0;

    if(ind == n) {
        // condition satisfied
        if(s == sum) return 1;
        // condition not satisfied
        else return 0;
    }


    s += arr[ind];

    int l = printS(ind+1,  s, sum, arr, n) ;

    s-= arr[ind];


    // not pick
    int r = printS(ind+1, s, sum, arr, n);

    return l + r;
}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {1, 2, 1};
    int n = 3;
    int sum = 2;
    cout << printS(0, 0, sum, arr, n);
```