SCALER

Backtracking

TABLE OF CONTENTS

Output based questions

```
int magicfun (int N){
    if (N==0) {return 0}
    else{
      return magicfun(N/2) * 10 + (N%2);
    }
}
```

10

mf (10) {
  return 10%2 + 10 * mf(5) {
    return 5%2 + 10 * mf(2) {
      return 2%2 + 10 * mf(1) {
        return 1%2 + 10 * mf(0) {
          return 0
        }
      }
    }
  }
}

1010

0 + 10 * 101 = 1010
1 + 10 * 10 = 101
0 + 10 * 1 = 10
1 + 10 * 0 = 1

```
void solve (N) {
    if (N == 0) return
    print (N)
    solve (N-1)
    print (N)
}
```

2

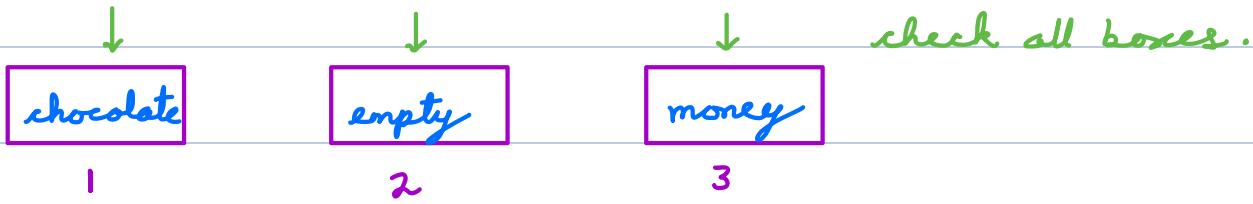
o/p → 2 1 1 2

solve (2) {
  print (2) ✓
  solve (1) {
    print (1) ✓
    solve (0) { return }
    print (1) ✓
  }
  print (2) ✓
}

Backtracking

Try all possibilities using recursion.

closed box



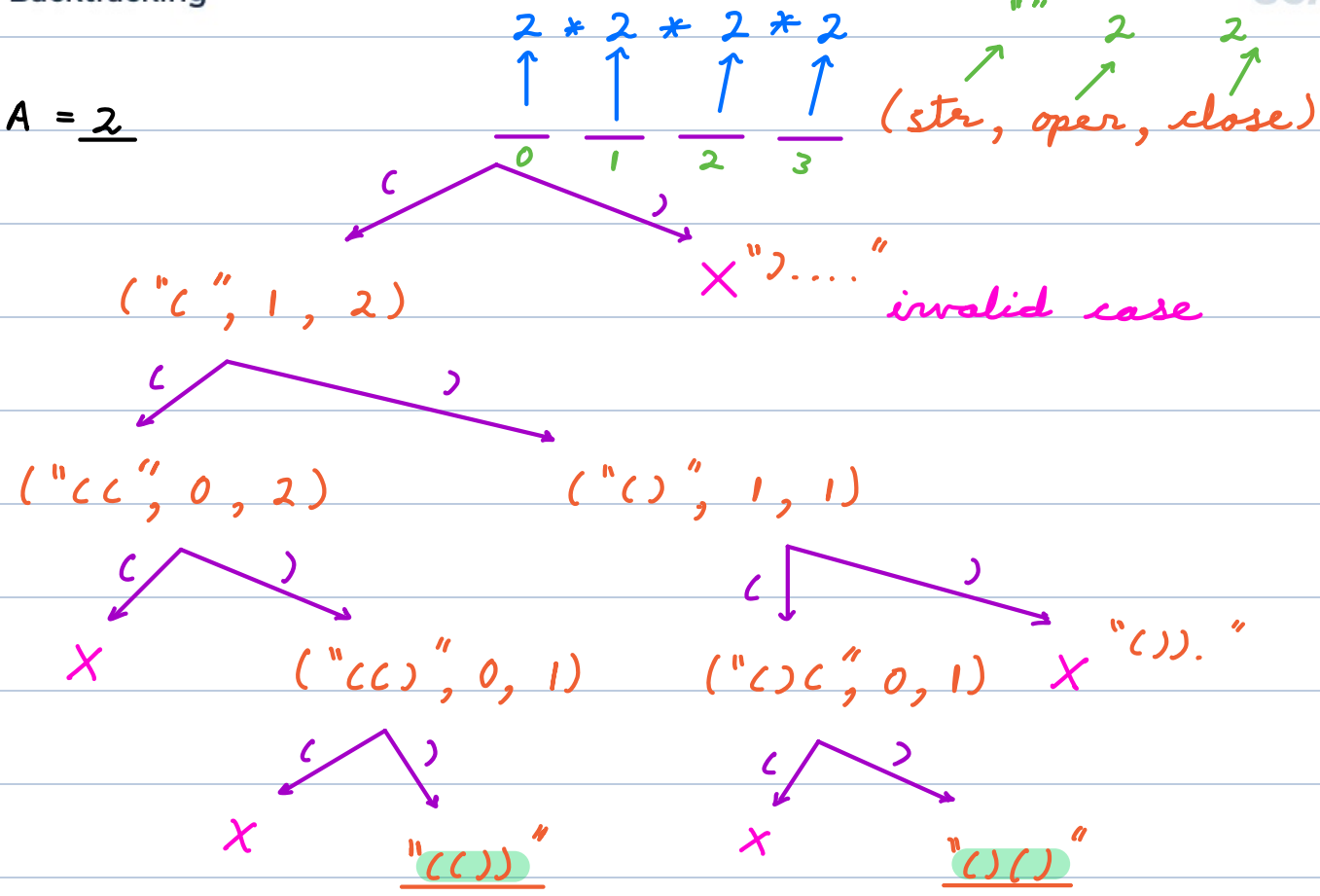
< *Question* > : Given an integer A pairs of parentheses, write a function to generate all combinations of well-formed parentheses of length 2*A.

A = 1 ()

A = 2 ()() , (()) ()()(X)

A = 3 ((())) , (()()) , (())() , ()(()) , ()()()

A = 2



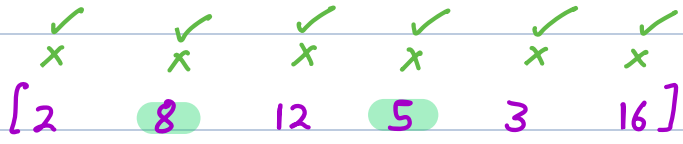

```
void solve(str, oper, close) {
    if (oper == 0 && close == 0) {
        print(str)
        return
    }
    if (oper > 0)   solve(str + '(', oper - 1, close)
    if (close > oper)  solve(str + ')', oper, close - 1)
}
```

(str: "", oper: A, close: A)

TC < $O(2^N)$ SC = $O(N)$

(small constraints)

Definition of Subset and Subsequences



subset = 2^N

same subsets

[8 5] → Subset → May or maynot be continuous.

[2 5 16] (Any possible selection)

[12]

[ ]

[5 8]

X

Subsequence → Subset where order of selected elements is maintained.

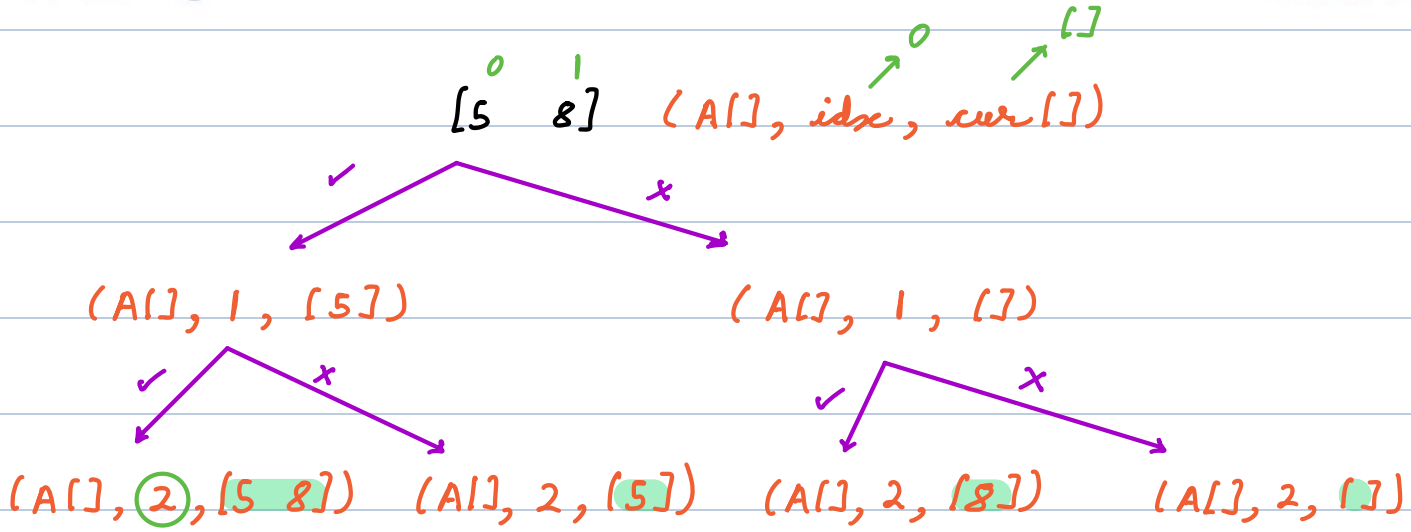
Q → Given an array of distinct integers, print all subset of the array.

A = [5 8] [ ] [5] [8] [5 8]

A = [1 2 3] [ ] [1] [2] [3]

[1 2] [1 3] [2 3]

[1 2 3]




```
Arraylist < Arraylist <Integer> > ans ;
void subset (A[], idx, cur[]) {
    if ( idx == A.length ) {   // basecase
        copy = new Arraylist (cur)
        ans . add (copy)
        return
    }
    cur . add (A[idx])          // do
    subset (A, idx+1, cur)      // select
    cur . removelast ()         // undo
    subset (A, idx+1, cur)      // reject
}
```

TC = $O(2^N)$ SC = $O(N)$

Data in every subset to be sorted → sort i/p.

H.W → Return ans in sorted order.

Problem

A popular Fitness app **FitBit**, is looking to make workouts more exciting for its users. The app has noticed that people get bored when the same exercises are shown in the same order every time they work out. To mix things up, **FitBit** wants to show all the different ways the exercises can be arranged so that each workout feels new.

Your challenge is to write a program for **FitBit** that takes a string **A** as input, where each character in the string represents a different exercise. Your program should then find and display all possible arrangements of these exercises.

Example:
A = Push-ups
B = Squats
C = Burpees
D = Planks

Then different ways of doing the exercise includes -
ABCD
ACBD
ADBC
ADCB
.

a b c

a b c
a c b
b a c
b c a
c a b
c b a

a b c d

↓ ↓ ↓ ↓

4 * 3 * 2 * 1 = 24 (4!)

Permutations

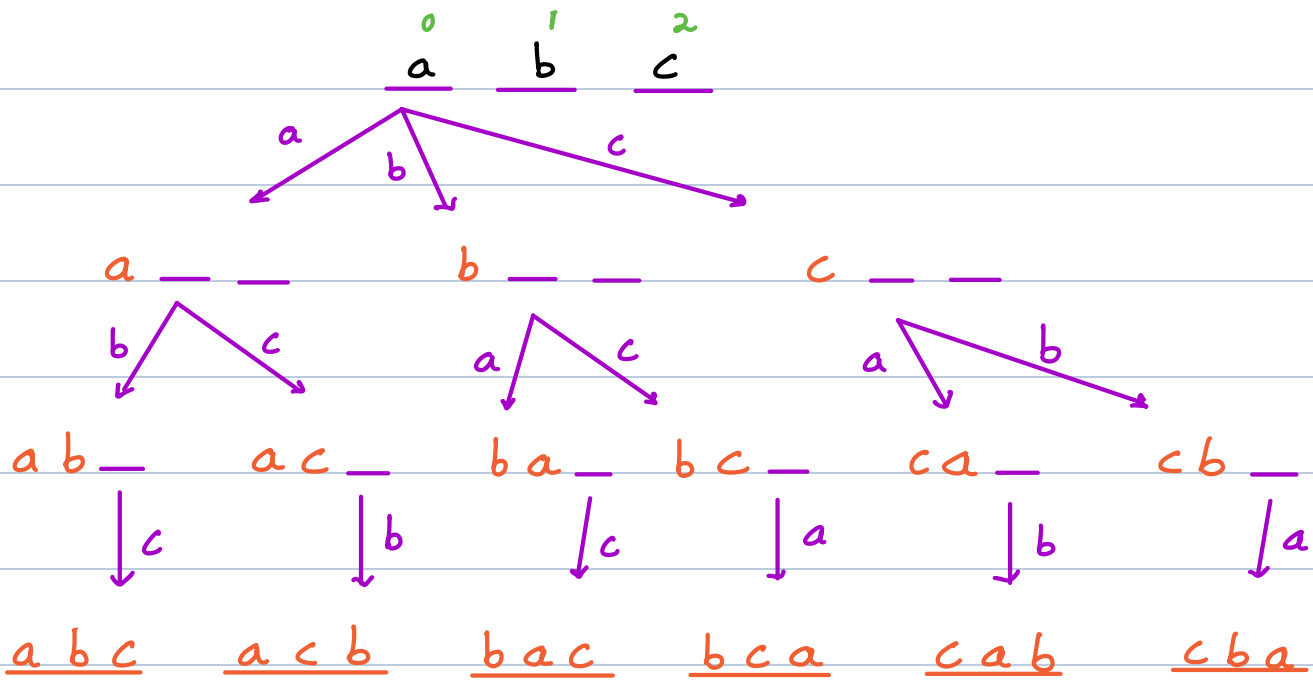
< *Question* > : Given a character array with distinct elements, print all permutations of it without modifying it.

A → [a b c]

a b c a c b

b a c b c a

c a b c b a



```
void permutation(A[], idx, cur[], vst[]) {        // idx: 0, cur: empty[], vst: ∀i, false
    if (idx == A.length) {                        // Base Case
        print(cur)
        return
    }
    for i → 0 to (N-1) {                          // All Possibilities
        if (!vst[i]) {                            // Valid Possibility
            vst[i] = true                         // Do
            cur[idx] = A[i]
            permutation(A, idx+1, cur, vst)       // Recursion
            vst[i] = false                        // Undo
        }
    }
}
```

TC = O(N!) SC = O(N)