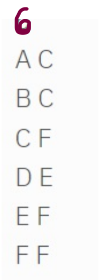
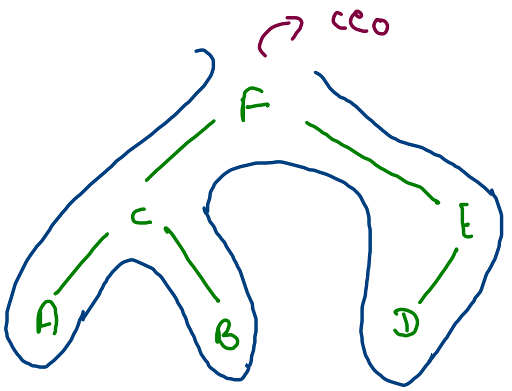
Number Of Employees Under Every Manager



```
6
A C
B C
C F
D E
E F
F F
```

ceo = F

emp -> manager



manager vs employees

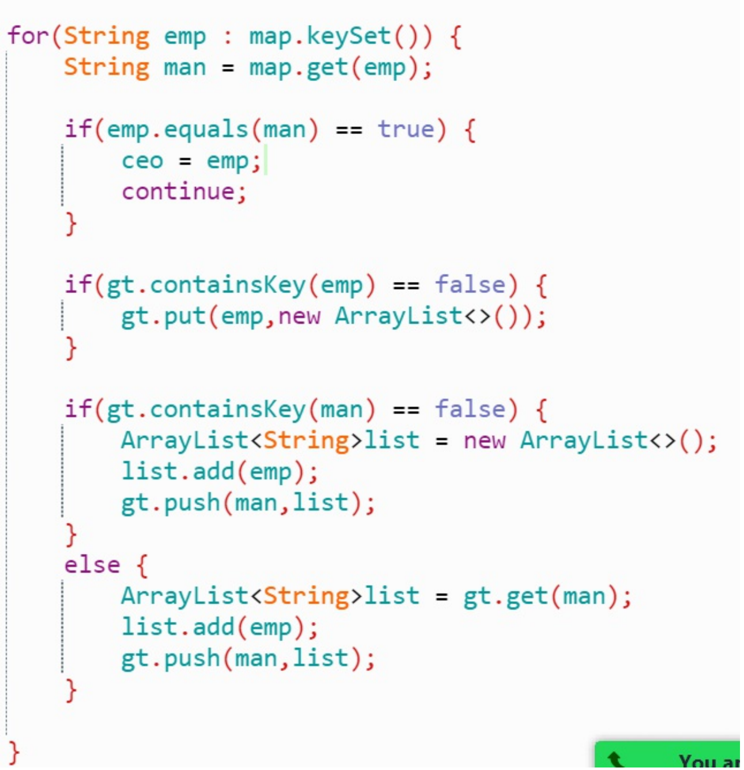
↓ └→ Generic tree

String vs ArrayList <String>

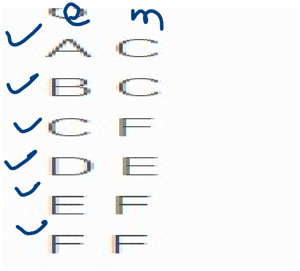
C — A, B
A
B
F - C, E
D
E — D

A - 0
B - 0
C - 2
D - 0
E - 1
F - 5

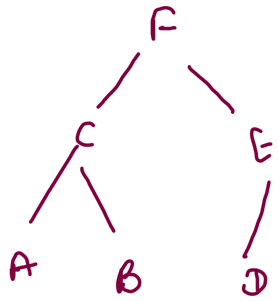
ceo = F



```
for(String emp : map.keySet()) {
    String man = map.get(emp);

    if(emp.equals(man) == true) {
        ceo = emp;
        continue;
    }

    if(gt.containsKey(emp) == false) {
        gt.put(emp,new ArrayList<>());
    }

    if(gt.containsKey(man) == false) {
        ArrayList<String>list = new ArrayList<>();
        list.add(emp);
        gt.push(man,list);
    }
    else {
        ArrayList<String>list = gt.get(man);
        list.add(emp);
        gt.push(man,list);
    }

}
```



	e	m
✓	A	C
✓	B	C
✓	C	F
✓	D	E
✓	E	F
✓	F	F



ceo

gt

A -> .

C -> A, B

B -> .

F -> C, E

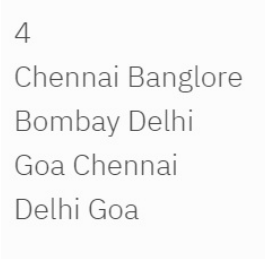
D -> .

E -> D

ceo = F

Find Itinerary From Tickets

→ A plan of journey



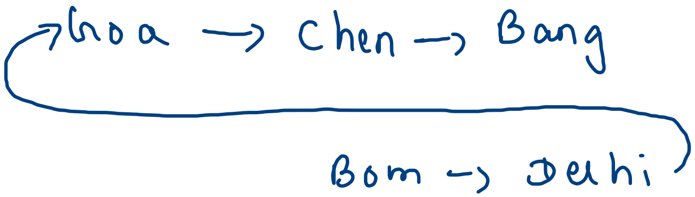
```
4
Chennai Banglore
Bombay Delhi
Goa Chennai
Delhi Goa
```

Chen -> Bangalore

Bom -> Delhi

Goa -> chen

Delhi -> Goa



Bombay -> Delhi -> Goa -> chennai -> Bangalore.

Linked list

↓

head

map.

String vs Boolean

Chen -> X

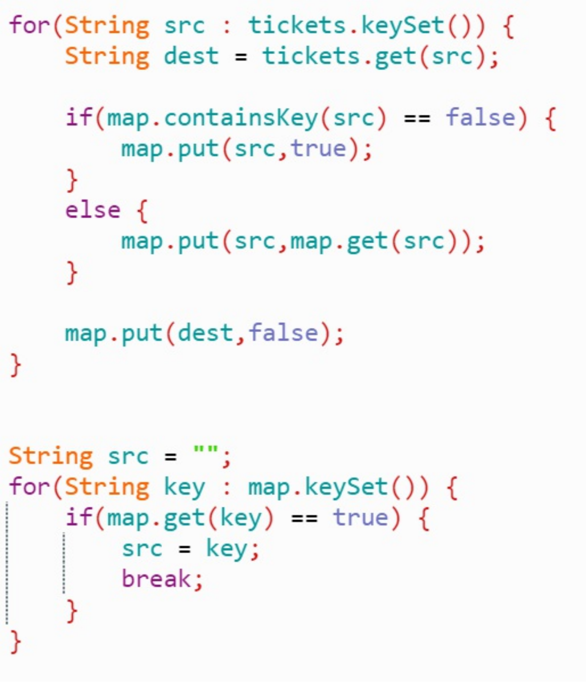
Bang -> X

Bom -> ✓

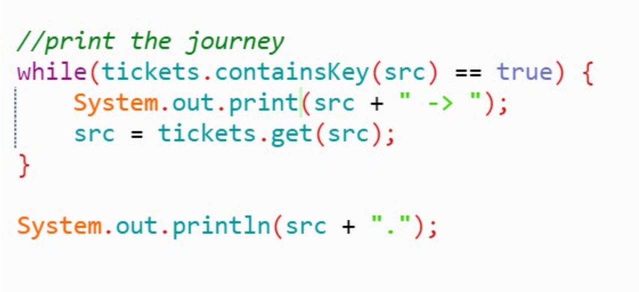
Delhi -> X

Goa -> X

src = Bombay

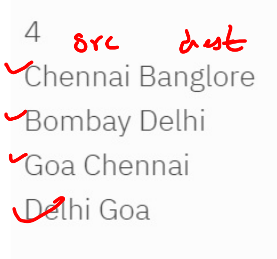


```java
for(String src : tickets.keySet()) {
    String dest = tickets.get(src);

    if(map.containsKey(src) == false) {
        map.put(src,true);
    }
    else {
        map.put(src,map.get(src));
    }

    map.put(dest,false);
}


String src = "";
for(String key : map.keySet()) {
    if(map.get(key) == true) {
        src = key;
        break;
    }
}
```



```java
//print the journey
while(tickets.containsKey(src) == true) {
    System.out.print(src + " -> ");
    src = tickets.get(src);
}

System.out.println(src + ".");
```

tickets



```
4  src   dest
✓Chennai Banglore
✓Bombay Delhi
✓Goa Chennai
✓Delhi Goa
```

chennai -> false

Bangalore -> False

Bombay -> true

Delhi -> false

Goa -> false

Bangalore

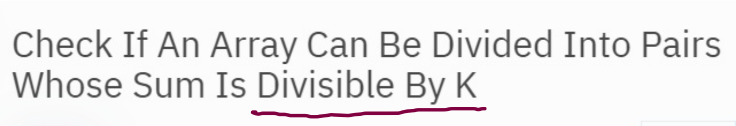
~~chennai~~

~~Goa~~

~~Delhi~~

src = ~~Bombay~~

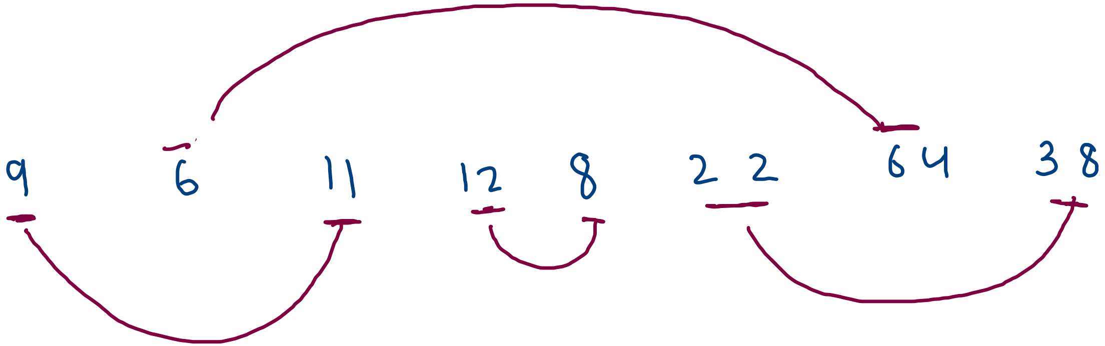
Bombay -> Delhi -> Goa -> chennai -> Bangalore .



Check If An Array Can Be Divided Into Pairs Whose Sum Is Divisible By K

rem

$k = 10$



T : $O(n)$

remainders

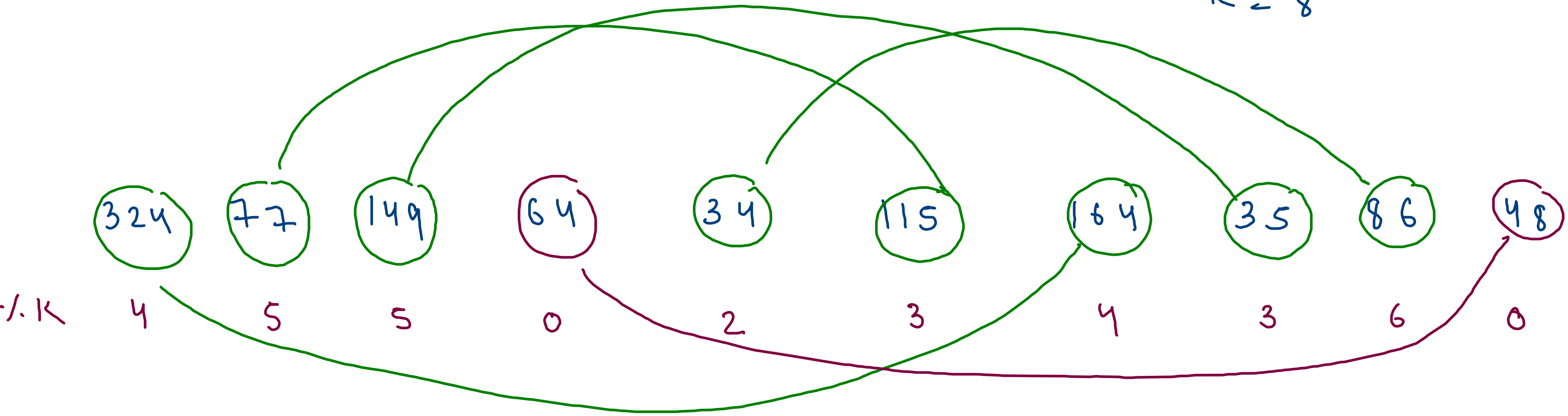
store

$n_1 \to kn + x$

$n_2 \to km + y$

$n_1 + n_2$ is div by k, $\boxed{y = k - x}$

K = 8



✓ 4 -> 2

5 -> 2

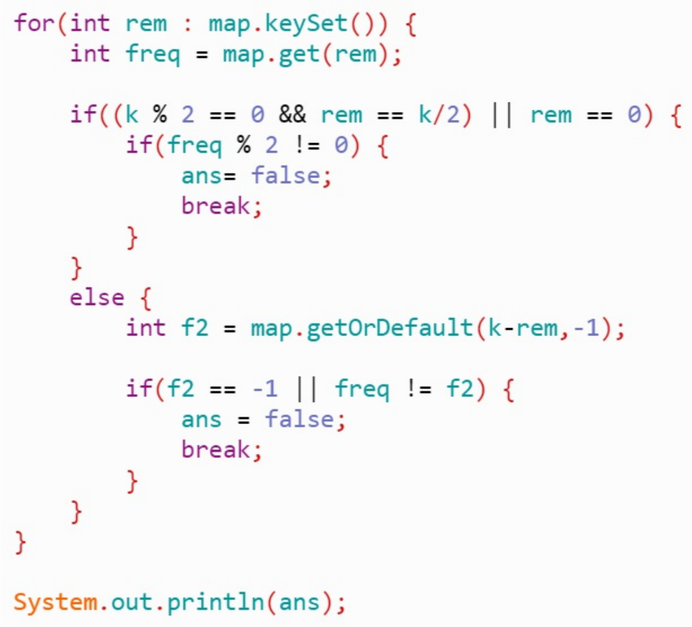
0 -> 2

3 -> 2

2 -> 2

6 -> 2

k = 10



```java
for(int rem : map.keySet()) {
    int freq = map.get(rem);

    if((k % 2 == 0 && rem == k/2) || rem == 0) {
        if(freq % 2 != 0) {
            ans= false;
            break;
        }
    }
    else {
        int f2 = map.getOrDefault(k-rem,-1);

        if(f2 == -1 || freq != f2) {
            ans = false;
            break;
        }
    }
}

System.out.println(ans);
```

30 67 10 90 42 89 29 21

0 7 0 0 2 9 9 1

2 -> 2

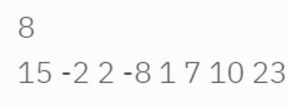
1 -> 2

7 -> 1

9 -> 2

0 -> 3

Largest Subarray With Zero Sum



```
8
15 -2 2 -8 1 7 10 23
```

T : o(n)

	15	-2	2	-8	1	7	10	23
	0	1	2	3	4	5	6	7
PS	15	13	15	7	8	15	25	48

len = ~~2~~ 5

0	->	-1
15	->	0
13	->	1
7	->	3
8	->	4
25	->	6
48	->	7

prefix sum vs it's first index

		10	9	-19	8	-4	6	-2	2	-11	1	8
	-1	0	1	2	3	4	5	6	7	8	9	10
ps	0	10	19	0	8	4	10	8	10	-1	0	8

len = ~~3~~ ~~5~~ ~~7~~ 10

0 ->	-1
10 ->	0
19 ->	1
8 ->	3
4 ->	4
-1 ->	8

ps vs it's first occ