1

```
t = input()
print(input().replace('.', t))
```

2

```
tables = {}
dishes = {}
for _ in range(int(input())):
    name, table, dish = input().split(',')
    table = int(table)
    if table not in tables:
        tables[table] = True
    if dish not in dishes:
        dishes[dish] = {}
    if table not in dishes[dish]:
        dishes[dish][table] = 0
    dishes[dish][table] += 1

keys = sorted(list(dishes.keys()))
tables = sorted(list(tables.keys()))
print('\t'.join(['Table'] + keys))
print()
for table in tables:
    ls = [str(table)]
    for dish in keys:
        if table not in dishes[dish]:
            ls.append('0')
        else:
            ls.append(str(dishes[dish][table]))
    print('\t'.join(ls))
    print()
```

3

bfs+剪枝

```
from queue import Queue

n, k = map(int, input().split())
q = Queue()
vis = {}

q.put((n, 0))
while True:
    x, t = q.get()
    if x == k:
        print(t)
        exit()

    if x in vis:
```

```
            continue
        vis[x] = True

        t += 1
        if x < k:
            q.put((2*x, t))
        q.put((x + 1, t))
        if x > 0:
            q.put((x - 1, t))
```

4

```
import sys

input = lambda : sys.stdin.readline().strip()

while n := int(input()):

    ls = []
    dic = {}
    vis = {}

    for _ in range(n):
        x, y = map(int, input().split())
        ls.append((x, y))
        dic[(x, y)] = True

    ls.sort()
    cnt = 0
    for i in range(n - 1):
        x1, y1 = ls[i]
        for j in range(i + 1, n):
            x2, y2 = ls[j]
            dx, dy = x2 - x1, y2 - y1

            x3, y3 = x1 - dy, y1 + dx
            x4, y4 = x2 - dy, y2 + dx

            key = tuple(sorted([(x1, y1), (x2, y2), (x3, y3), (x4, y4)]))
            if (x3, y3) in dic and (x4, y4) in dic and key not in vis:
                cnt += 1
                vis[key] = True
    print(cnt)
```

5

拓扑排序变形（9也可以过）

```
from heapq import *
import sys

input = lambda: sys.stdin.readline().strip()

for _ in range(int(input())):
```

```
    n = int(input())

    names = []
    edges = {i: {} for i in range(n)}
    hated = {i: {} for i in range(n)}
    in_degree = [0]*n
    cnt = 0

    for i in range(n):
        ls = input().split()
        names.append(ls[0])
        for t in ls[1:]:
            t = int(t) - 1
            edges[t][i] = hated[i][t] = True
            in_degree[i] += 1

    heap = []
    for i in range(n):
        if in_degree[i] == 0:
            heap.append(i)
    heapify(heap)
    ls = []
    inside = [False]*n
    while cnt < n:
        idx = heappop(heap)
        ls.append(names[idx])
        cnt += 1
        for i in hated[idx]:
            cnt -= 1
            for j in edges[i]:
                in_degree[j] += 1
            inside[i] = False
        for i in edges[idx]:
            in_degree[i] -= 1

        for i in hated[idx]:
            if in_degree[i] == 0 and not inside[i]:
                inside[i] = True
                heappush(heap, i)
        for i in edges[idx]:
            if in_degree[i] == 0 and not inside[i]:
                inside[i] = True
                heappush(heap, i)
    print(len(ls))
    print(*ls)
```

6

二分查找dfs

```
p, q, x, y = map(int, input().split())
dp = {q: 0}
mid = 0
```

```
def dfs(p, cnt):
    if cnt > mid:
        return False
    if p > q and (p - q)//x + cnt > mid:
        return False
    elif p > q and (p - q)//x + cnt <= mid and (p - q) % x == 0:
        return True

    if p == q:
        return True

    cnt += 1
    if p > x and dfs(p - x, cnt):
        return True
    if dfs(p*y, cnt):
        return True
    return False


mid = 52
if not dfs(p, 0):
    print('Failed')
    exit()

l, r = 0, 52
while l < r:
    mid = (l + r)//2

    if dfs(p, 0):
        r = mid
    else:
        l = max(mid, l + 1)
print(r)
```

7

二分查找

```
n = int(input())
times = [int(input()) for _ in range(n)]
m = int(input())

l, r = 0, sum(times) + 1


def judge(t):
    i = 0
    cnt = 0
    while i < n:
        total = 0
        max_today = 0
        while i < n and total <= t:
            total += times[i]
            max_today = max(max_today, times[i])
            i += 1
```

```
        total -= max_today

        while i < n:
            if total + times[i] <= t:
                total += times[i]
                i += 1
            else:
                break
        cnt += 1

    return cnt


while l < r:
    mid = (l + r) // 2
    cnt = judge(mid)

    if cnt <= m:
        r = mid
    else:
        l = max(mid, l + 1)

print(r)
```

8

和这道爆了。”输入包含若干银河的描述“以为是多组输入，价值理解反了，被硬控30分钟。

```
from queue import Queue

n = int(input())

values = []
names = []
ways = []
name_to_idx = {}
for i in range(n):
    t = input().split()
    if len(t) == 2:
        t.append('')
    name, value, way = t

    names.append(name)
    values.append(int(value[0] + value[2:]))
    ways.append(way)
    name_to_idx[name] = i

real_values = []
for i in range(n):
    q = Queue()
    q.put((i, 0))
    found = False
    vis = {}
    while not found and not q.empty():
```

```
        idx, distance = q.get()
        if idx in vis:
            continue
        vis[idx] = True

        for name in ways[idx]:
            if name == '*':
                found = True
                real_values.append((values[i]*0.95**distance, names[i]))
                break
            q.put((name_to_idx[name], distance + 1))

    if not found:
        real_values.append((0, names[i]))

real_values.sort(key=lambda t: (-t[0], t[1]))
print(real_values[0][1])
```