1

```
for _ in range(int(input())):
    s=list(map(int,input()))
    n=len(s);ans=temp=1
    for i in range(1,n):
        if s[i]>=s[i-1]:
            temp+=1
        else:
            ans=max(ans,temp)
            temp=1
    ans=max(ans,temp)
    print(ans)
```

3

```
a=[];s='([{'
dic={')':'(',']':'[','}':'{'}
flag=0
for i in input():
    if i in s:
        a.append(i)
    elif i in dic:
        if a[-1]!=dic[i]:
            print('ERROR');exit()
        a.pop()
        if a: flag=1
print('YES' if flag else 'NO')
```

4

```
n,m=map(int,input().split())
dp1=[0]*(n+1)
dp2=[0]*(n+1)
for i in range(1,n+1):
    a,b=map(int,input().split())
    dp1[i]=max(dp1[i-1],dp2[i-1]-m)+a
    dp2[i]=max(dp1[i-1]-m,dp2[i-1])+b
print(max(dp1[n],dp2[n]))
```

6

```
class UnionFind:
    def __init__(self,n):
        self.p=list(range(n))
        self.h=[0]*n
    def find(self,x):
        if self.p[x]!=x:
            self.p[x]=self.find(self.p[x])
        return self.p[x]
    def union(self,x,y):
        rootx=self.find(x)
```

```
        rooty=self.find(y)
        if rootx!=rooty:
            self.p[rootx]=rooty
n,m=map(int,input().split())
uf=UnionFind(n)
for _ in range(m):
    x,y=map(int,input().split())
    uf.union(x-1,y-1)
print(len(set([uf.find(i) for i in range(n)])))
```

7

```
from collections import deque
def check(a,b):
    i=0
    while a[i]==b[i]:
        i+=1
    if a[i+1:]==b[i+1:]:
        return True
    return False
def bfs(x,step):
    q=deque([(x,step)])
    while q:
        x,step=q.popleft()
        if check(x,y):
            return step+1
        for i in range(len(dic)):
            if i not in vis and check(x,dic[i]):
                q.append((dic[i],step+1))
                vis.add(i)
    return 0
x,y=input().split()
dic=list(input().split())
vis=set()
print(bfs(x,1))
```

8

```
class Node:
    def __init__(self,val):
        self.val=val
        self.left=None
        self.right=None
def check(x):
    if '0' not in x:return 'T'
    if '1' not in x:return 'Q'
    return 'E'
def build(s):
    root=Node(check(s))
    if len(s)>1:
        n=(len(s)+1)//2
        root.left=build(s[:n])
        root.right=build(s[n:])
    return root
```

```
def post(root):
    if not root:return ''
    return post(root.left)+post(root.right)+root.val
for _ in range(int(input())):
    s=input()
    root=build(s)
    print(post(root))
```

9

```
n=int(input())
graph=[[] for _ in range(n+1)]
children=[[] for _ in range(n+1)]
for _ in range(n-1):
    u,v=map(int,input().split())
    graph[u].append(v)
    graph[v].append(u)
cnt=[1]*(n+1);vis=[0]*(n+1)
def dfs(u):
    vis[u]=1
    for v in graph[u]:
        if not vis[v]:
            cnt[u]+=dfs(v)
            children[u].append(v)
    return cnt[u]
dfs(1);num=0
for i in range(1,n+1):
    temp=0
    if cnt[i]==1:continue
    for j in children[i]:
        if cnt[j]==1:temp+=1
    if temp%2==0:num+=1
print((n-num)//2)
```

10

```
from heapq import heappop,heappush
n,m=map(int,input().split())
graph=[[-1]*n for _ in range(n)]
for _ in range(m):
    u,v,w=map(float,input().split())
    u,v=int(u),int(v)
    graph[u][v]=graph[v][u]=w
vis=[0]*n;q=[(0,0,-1)]
ans=0;pairs=[]
while q:
    w,u,pre=heappop(q)
    if vis[u]:continue
    ans+=w;vis[u]=1
    for v in range(n):
        if not vis[v] and graph[u][v]!=-1:
            heappush(q,(graph[u][v],v,u))
    if u>pre:u,pre=pre,u
    if u!=-1:pairs.append((u,pre))
```

```
if len(pairs)==n-1:
    print(f'{ans:.2f}')
    for i in pairs:print(*i)
else:print("NOT CONNECTED")
```

11

没想出来，拜读了邓佬的代码

12

注：这是笨方法，巧方法见github

```
class Node:
    def __init__(self,val):
        self.val=val
        self.left=None
        self.right=None
def insert(root,num):
    if not root:return Node(num)
    if num<root.val:
        root.left=insert(root.left,num)
    else:
        root.right=insert(root.right,num)
    return root
def post(root):
    if not root:return []
    return post(root.left)+post(root.right)+[root.val]
n=int(input())
a=list(map(int,input().split()))
root=Node(a[0])
for i in a[1:]:
    insert(root,i)
print(*post(root))
```