Editorial :
The score is number of set bits in xor of two arrays to get score equal to K we want k set bits and (n-k) reset bits in xor .
There are $\binom{n}{k}$ ways to do that .

If we want i'th bit to set in xor there are two ways
1. arr[i]=1 and brr[i]=0
2. arr[i]=0 and brr[i]=1

If we want i'th bit to reset in xor there are two ways
1. arr[i]=0 and brr[i]=0
2. arr[i]=1 and brr[i]=1

Answere = $\binom{n}{k}2^n$

Precompute all the factorial and inverse factorial under a MOD.
Complexity :
Per test case : O(log(n))
Pre computation :O(MAX log(MAX))
Total :O(MAX log(MAX))

CODE :

```
MOD=10**9+7
##binary exponentiation to compute a**b(modulo MOD)
def binary_exp(a,b,MOD):
    ans=1
    while b:
        if b&1:
            ans*=a
            ans%=MOD
        a*=a
        a%=MOD
        b//=2
    return ans

fac=[1]*(10**5+1)
##computing factorial under MOD
for i in range(2,10**5+1):
    fac[i]=fac[i-1]*i
    fac[i]%=MOD

##computing factorial inverse under MOD
facinv=[binary_exp(i,MOD-2,MOD) for i in fac]
```

```
for _ in range(int(input())):
    n,k=map(int,input().split())
    res=fac[n]*facinv[n-k]*facinv[k]
    res%=MOD
    res*=binary_exp(2,n,MOD)
    res%=MOD
    print(res)
```