一、简答

1. 转移矩阵为

$$\begin{bmatrix} 7 & 0 & 3 & 4 & 2 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

二、代码及注释

B.自然数幂和

题目大意

给定 n 和 k，计算 $\sum_{i=1}^{n} i^k$ 对 10^9+7 取模的结果。

解法

令$s_n=\sum_{i=1}^{n} i^k$,那么$s_n=s_{n-1}+n^k$
对n^k作二项式展开得
$n^k=C_k^0(n-1)^k+C_k^1(n-1)^{k-1}+\cdots+C_k^k(n-1)^0$
同理，
$n^{k-1}=C_{k-1}^0(n-1)^{k-1}+C_{k-1}^1(n-1)^{k-2}+\cdots+C_{k-1}^{k-1}(n-1)^0$
观察得
我们的状态可以设定为 $\begin{bmatrix} s_n \\ n^k \\ n^{k-1} \\ \vdots \\ 1 \end{bmatrix}$

推得转移矩阵，我们如何构造它呢？首先转移矩阵是只由 k 确定，每个元素是 k 的组合数，可以通过 dp 推得，转移方程为 $C_n^m=C_{n-1}^m+C_{n-1}^{m-1}$.
之后按照一定规律填充转移矩阵即可。
dp 的起始状态为 $[1,1,...,1]^T$ ，即 n = 1，最终答案为转移矩阵的 (n - 1) 次幂的第一行求和。

时间复杂度

$O(k^3logn)$, 对转移矩阵(k大小方阵)做 $logn$ 次矩阵乘法。

代码

```cpp
#include<iostream>
#include<vector>
using namespace std;
using ll = long long;

const int Mod = 1e9 + 7;
int C[15][15];

struct mat
{
        int n;
        vector<vector<long long>> x;

        // 构造函数
        explicit mat(int sz) : x(sz, vector<long long>(sz)) { n = sz; }

        // 单位矩阵
        void emat()
        {
                for (int i = 0; i < n; i++) {
                        x[i][i] = 1;
                }
        }

        // 重载乘号
        mat operator*(const mat &a) const
        {
                mat res(n);
                for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++) {
                                for (int k = 0; k < n; k++) {
                                        res.x[i][j] = (res.x[i][j] + x[i][k] * a.x[k][j]) % Mod;
                                }
                        }
                }
                return res;
        }
};

// 矩阵快速幂
mat matqp(mat a, long long b)
{
        mat res(a.n);
        res.emat();
        while (b)
        {
                if (b & 1) res = res * a;       // 重载*乘法
                a = a * a;
                b >>= 1;
        }
```

```
        return res;
}

void solve() {
    int n, k;
    cin >> n >> k;

    mat t(k + 2);

    // 组合数计算 C00 = 0
    for (int i = 0; i <= k; i++) C[i][0] = 1;
    for (int i = 1; i <= k; i++) {
        for (int j = 1; j <= i; j++) {
            C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
            // cout << "C[" << i << "][" << j << "] = " << C[i][j] << ' ';
        }
        // cout << '\n';
    }

    // 构造转移矩阵
    t.x[0][0] = 1;
    for (int i = 1; i < k + 2; i++) {
        t.x[0][i] = C[k][i - 1];
    }

    for (int i = 1; i <= k; i++) {
        for (int j = i; j < k + 2; j++) {
            t.x[i][j] = C[k - i + 1][j - i];
        }
    }
    t.x[k + 1][k + 1] = 1;

    mat res = matqp(t, n - 1);
    // 初始状态s1=[1,1,...,1]^T
    ll ans = 0;
    for (int i = 0; i < k + 2; i++) {
        ans = (ans + res.x[0][i]) % Mod;
    }
    cout << ans << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t; cin >> t;
    while (t--)
        solve();
    return 0;
}
```