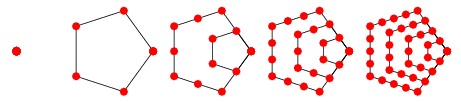
[五边形数定理](http://blog.csdn.net/acdreamers/article/details/12259815)

**设第n个五边形数为，那么，即序列为：1, 5, 12, 22, 35, 51, 70, ...**

**对应图形如下：**

****

**设五边形数的生成函数为，那么有：**

****

**以上是五边形数的情况。下面是关于五边形数定理的内容：**

**五边形数定理是一个由欧拉发现的数学定理，描述欧拉函数展开式的特性。欧拉函数的展开式如下：**





**欧拉函数展开后，有些次方项被消去，只留下次方项为1, 2, 5, 7, 12, ...的项次，留下来的次方恰为广义五边形数。**

**五边形数和分割函数的关系**

**欧拉函数的倒数是分割函数的母函数，亦即：**

**其中为k的分割函数。**

**上式配合五边形数定理，有：**



**在 n>0 时，等式右侧的系数均为0，比较等式二侧的系数，可得**



**因此可得到分割函数p(n)的递归式：**

**例如n=10时，有：所以，通过上面递归式，我们可以很快速地计算n的整数划分方案数p(n)了。**

1. **const** **int** N=100005;
2. **const** LL MOD=1000000007;
4. LL ans[N],tmp[N];
6. **void** Init()
7. {
8. **int** t=1000;
9. **for**(**int** i=-1000;i<=1000;i++)
10. tmp[i+t]=i\*(3\*i-1)/2;
11. ans[0]=1;
12. **for**(**int** i=1;i<N;i++)
13. {
14. ans[i]=0;
15. **for**(**int** j=1;j<=i;j++)
16. {
17. **if**(tmp[j+t]<=i)
18. {
19. **if**(j&1)  ans[i]+=ans[i-tmp[j+t]];
20. **else**     ans[i]-=ans[i-tmp[j+t]];
21. }
22. **else** **break**;
23. ans[i]=(ans[i]%MOD+MOD)%MOD;
24. **if**(tmp[t-j]<=i)
25. {
26. **if**(j&1) ans[i]+=ans[i-tmp[t-j]];
27. **else**    ans[i]-=ans[i-tmp[t-j]];
28. }
29. **else** **break**;
30. }
31. ans[i]=(ans[i]%MOD+MOD)%MOD;
32. }
33. }
34. **int** main()
35. {
36. **int** t,n;
37. Init();
38. cin>>t;
39. **while**(t--)
40. {
41. cin>>n;
42. cout<<ans[n]<<endl;
43. }
44. **return** 0;
45. }

**题意：问一个数n能被拆分成多少种方法，且每一种方法里数字重复个数不能超过k（等于k）。**

**分析递推式为**



1. **const** **int** N = 100005;
2. **const** **int** MOD = 1000000007;
4. **int** dp[N];
6. **void** Init()
7. {
8. dp[0] = 1;
9. **for**(**int** i=1;i<N;i++)
10. {
11. dp[i] = 0;
12. **for**(**int** j=1;;j++)
13. {
14. **int** t = (3\*j-1)\*j / 2;
15. **if**(t > i) **break**;
16. **int** tt = dp[i-t];
17. **if**(t+j <= i) tt = (tt + dp[i-t-j])%MOD;
18. **if**(j&1) dp[i] = (dp[i] + tt)%MOD;
19. **else**    dp[i] = (dp[i] - tt + MOD)%MOD;
20. }
21. }
22. }
24. **int** Work(**int** n,**int** k)
25. {
26. **int** ans = dp[n];
27. **for**(**int** i=1;;i++)
28. {
29. **int** t = k\*i\*(3\*i-1) / 2;
30. **if**(t > n) **break**;
31. **int** tt = dp[n-t];
32. **if**(t + i\*k <= n) tt = (tt + dp[n-t-i\*k])%MOD;
33. **if**(i&1) ans = (ans - tt + MOD)%MOD;
34. **else**    ans = (ans + tt)%MOD;
35. }
36. **return** ans;
37. }
39. **int** main()
40. {
41. Init();
42. **int** n,k,t;
43. scanf("%d",&t);
44. **while**(t--)
45. {
46. scanf("%d%d",&n,&k);
47. printf("%d\n",Work(n,k));
48. }
49. **return** 0;

}