**有奖答题**

**题目描述：**

公元2222年，地球人与花之星球人建立了友好的星际关系，花之星球的旅游社抓住了这个机遇，在地球上做广告吸引地球游客去当地旅游。今年小明刚刚高考考完看到广告后便决定去花之星球过暑假。但是一次星际穿越的费用非常贵约为166,666,000信用点！这可怎么办呢？小明在绝望中突然发现旅游社有个活动，参加该活动，胜利者可以免费领取两张花之星球往返飞船票。让我们来看看活动的内容吧：

！！你离花之星球只差一道题的距离！！

花花旅行社：2222年6月30日在大广场举行有奖答题活动，第一位答出题目者将获得由花花旅行社免费赠送的两张银河5号超光速飞船的“地球-花之星球”往返船票。题目如下：

在花之星球上，所有的花都有雄蕊、花瓣和萼片，并且每种花雄蕊的数目、花瓣的数目和萼片的数目都遵循一种特殊规律。在花之星球上的任何一朵花，如果有n片花瓣，m片萼片，那么雄蕊的数目必定是一个m位整数，并且它每一位的n次方之和刚好等于这个整数。同时如果存在一个m位整数，并且它每一位的n次方之和刚好等于这个整数，那么这样的花也一定存在。在有奖答题活动当天我们会随机的给出几个m,n你需要在最短的时间内回答：花之星球上是否存在花瓣数为n花萼数为m的花，如果存在则说出一共有几种这样的花。

POST:当然花之星球使用的是3进制数,而且每一种已经发现花都可以在百科全书上找到，为了避免作弊你需要按照十进制作答。

小明看见了题目之后头晕眼花，就向班里的考神求助，考神看了看之后觉得可以用计算机来完成这件事，便让小明来找作为学神的你，让你帮忙写一个能用尽可能短的时间回答旅行社出的问题的程序，小明还答应得奖之后将其中一张船票送你。你其实也很想到地球外面旅游一下，便欣然答应了。

**输入：**

输入包含多组数据，每组数据占一行并包含两个整数m n。 m<32是萼片数，n<15是花瓣数

**输出：**

对于每组数据输出一个整数，每组输出占一行。

**输入示例：**

0 0

0 1

1 1

11 11

12 12

**输出示例：**

0

0

9

8

0

**题目大意：**

题目要求求出有多少个n位整数，每一位的m次方之和等于这个数。n<=0.m<=0时答案为0。

**题解：**

那么如果我们从10000……000一直计算到9999……999的话时间复杂度为O(10^n)这样肯定不行，观察题目可以发现，如果几个数中每种数码出现的次数相同，那么每一位的m方和也是一样的，因此对于每种数码出现的次数都相同的一类数，我们只需要计算其中一个数即可，为了方便我们只计算这样的数：它的前一位数码不大于后一位数码，如123456789，11112223334999。因为所有的数都可以通过改变数码的位置来满足这个要求并且满足这个要求的数是唯一的，所以对于任意一个数，我们都可以找到且仅能找到一个这样的数，使两个数中每种数码出现的次数相同。只计算这样的数的时间复杂度为O((n + 9)!/(9! n!))=O(n^9)。

POST：为了更快的速度，你还需要进行常数优化。

**测试数据：**

-8 0

0 -2

0 0

1 0

0 1

1 1

30 1

3 3

4 4

5 4

5 5

5 6

6 6

7 7

7 8

8 8

9 9

10 10

11 11

13 12

14 14

31 14

26 13

29 13

25 14

**测试答案：**

0

0

0

0

0

9

1

4

3

2

3

1

1

4

1

3

4

1

8

1

1

0

0

0

0

**标程：**

#include<stdio.h>

#define \_KK 32 //最大位数 //\_K不得小于\_II,否则数组满出

#define \_II 32 //最大次方数

#define \_JJ 10 //10进制

struct lll{//long long long 第一位数码是x[1]

int x[\_KK];

};

struct lll multi[\_II][\_JJ]; /\* j^i次方表 \*/

void checklll(struct lll \*a) /\* 进位 \*/

{

int k;

for(k=0;k<\_KK-1;k++){

a->x[k+1]+=a->x[k]/\_JJ;

a->x[k]%=\_JJ;

if(a->x[k]<0){

a->x[k]+=\_JJ;

a->x[k+1]--;

}

}

}

int fill\_table() /\* 填充次方表 \*/

{

int i,j,k;

for(j=0;j<\_JJ;j++){

for(k=0;k<\_KK;k++) multi[0][j].x[k]=0;

multi[0][j].x[1]=1;

}

for(i=1;i<\_II;i++){

for(j=0;j<\_JJ;j++){

for(k=0;k<\_KK;k++) multi[i][j].x[k]=multi[i-1][j].x[k]\*j;

checklll(&multi[i][j]);

}

}

}

int I\_,K\_;//输入 I\_ 次方 K\_ 位数

int anss;//结果的个数

struct lll now;//当前数字

int na[\_JJ];//各位数码计数

struct lll s;/\* 次方和 \*/

int nb[\_JJ];//各位数码计数

void f3()//检查

{

int j,k;

checklll(&s); // /\* 求和后进位

for(j=0;j<\_JJ;j++) nb[j]=0;

for(k=1;k<=K\_;k++) nb[s.x[k]]++;//各位数码计数

int mark=0;//检测标记

for(j=0;j<\_JJ;j++) if(na[j]==nb[j]) mark++;// /\* 循环最里层 \*/

if(mark==\_JJ&&(s.x[K\_]!=0)) //大致判断

{

anss++;

for(k=K\_+1;k<\_KK;k++)//高位检测 ~\_~

{

if(s.x[k]>0){

anss--;

break;

}

}

}

}

int f2(int i)//遍历

{

int k;

if(i>K\_) f3();

else

{

for(now.x[i]=now.x[i-1];now.x[i]<\_JJ;now.x[i]++)//遍历数码

{

na[now.x[i]]++;

for(k=0;k<\_KK;k++) s.x[k]+=multi[I\_][now.x[i]].x[k]; //累加次方和

f2(i+1);//下一位

for(k=0;k<\_KK;k++) s.x[k]-=multi[I\_][now.x[i]].x[k]; //回退次方和

na[now.x[i]]--;

}

}

}

int f1()//启动

{

int j,k;

anss=0;

for(j=0;j<\_JJ;j++) na[j]=0; //初始化

for(k=0;k<\_KK;k++) s.x[k]=0;

now.x[0]=0;

f2(1);

return 0;

}

int main()

{

fill\_table();

while(~scanf("%d %d",&I\_,&K\_))

{

if(I\_<=0||K\_<=0) printf("0\n");

else{

f1();

printf("%d\n",anss);

}

}

return 0;

}