题目描述：

某组织举行会议，来了多个代表团同时到达，接待处只有一辆汽车，可以同时接待多个代表团，为了提高车辆利用率，请帮接待员计算可以坐满车的接待方案，输出方案数量。

约束：

1、一个团只能上一辆车，并且代表团人数（代表团数量小于 30，每个代表团人数小于 30）

小于汽车容量（汽车容量小于 100）

2、需要将车辆坐满

输入描述：

第一行 代表团人数，英文逗号隔开，代表团数量小于 30，每个代表团人数小于 30

第二行 汽车载客量 ，汽车容量小于 100

输出描述：

坐满汽车的方案数量

如果无解输出 0

补充说明：

各代表团人数 5,4,2,3,2,4,9

汽车载客量 10

输出 4

解释 以下几种方式都可以坐满车，所以，优先接待输出为 4

[2, 3, 5]

[2, 4, 4]

[2, 3, 5]

[2, 4, 4]

示例 1

输入：

```
5,4,2,3,2,4,9
10
```

输出：

```
4
```

说明：

解释 以下几种方式都可以坐满车，所以，优先接待输出为 4

[2, 3, 5]

[2, 4, 4]

[2, 3, 5]

[2, 4, 4]

```
#include <bits/stdc++.h>

using namespace std;

long long i, j, k, n, m, t, T, f, len, co;
char s[1001];
long long a[31];
long long dp[31][101];

int main() {

	scanf("%s", s + 1);
	scanf("%lld", &n);
	for (len = strlen(s + 1), s[len + 1] = ',', i = 1; i <= len + 1; i++)
	{
		if (s[i] == ',') a[++co] = t, t = 0;
		else t = t * 10 + s[i] - '0';
	}

	dp[1][a[1]] = 1;
	for (i = 2; i <= co; i++) {
		for (j = 1; j <= n; j++) {
			dp[i][j] = max(dp[i][j], dp[i - 1][j]);
			if (j - a[i] >= 0) {
				dp[i][j] += dp[i - 1][j - a[i]];
			}
```

```
            }
            dp[i][a[i]]++;
        }

        printf("%lld", dp[co][n]);

        return 0;
}
//1,2,1,1
//2
//
//4
```