#include<cstdio>

#include<cstring>

using namespace std;

typedef long long ll;

const int N = 15;

ll f[N][2][N][2];

int num[N]; //num来存这个数每个位子上的数码

/\*

记忆化搜索。

len是当前为从高到低第几位。issmall表示当前位是否和num[len]相等，0是相等，1是不相等。

sum表示当前数字出现的次数。zero表示之前是否是前导0。d是当前在算的数码。

\*/

ll dfs(int len, bool issmall, int sum, bool zero, int d)

{

ll ret = 0;

if (len == 0) return sum; //边界条件

if (f[len][issmall][sum][zero] != -1) return f[len][issmall][sum][zero]; //记忆化

for (int i = 0; i < 10; i ++){

if (!issmall && i > num[len]) break;

/\*

由于我们是从高位到低位枚举的，所以如果之前一位的数码和最大数的数码相同，这一位就只能枚举到num[len]；

否则如果之前一位比最大数的数码小，那这一位就可以从0~9枚举了。

\*/

ret += dfs(len-1, issmall || (i<num[len]), sum+((!zero || i) && (i==d)), zero && (i == 0), d);

/\*

继续搜索，数位减一，issmall的更新要看之前有没有相等，且这一位有没有相等；

sum的更新要看之前是否为前导0或者这一位不是0；

zero的更新就看之前是否为前导0且这一位继续为0；

d继续传进去。

\*/

}

f[len][issmall][sum][zero] = ret;

//记忆化，把搜到的都记下来

return ret;

}

ll solve(ll x, int d)

{

int len = 0;

while (x){

num[++ len] = x%10;

x /= 10;

} //数字转数位

memset(f, -1, sizeof f); //初始化

return dfs(len, 0, 0, 1, d); //开始在第len位上，最高位只能枚举到num[len]所以issmall是0，sum=0，有前导0。

}

int main()

{

ll a, b; //注意都要开long long

scanf("%lld%lld", &a, &b);

for (int i = 0; i < 10; i ++)

printf("%lld%c", solve(b, i)-solve(a-1, i), i == 9 ? '\n' : ' ');

return 0;

}