#include<bits/stdc++.h>
  
using namespace std;
  
  
int n; // n种面值
  
int m; // 最多允许贴m张
  
vector<int> result; // 最终结果
  
vector<int> temp; // 记录已经确定下来的面值
  
int maxRe = 0; //最大邮资
  
  
//判断由已选的前cur个面值,在张数不超过m的情况下能否贴出该邮资re
  
//回溯法判断 re:当前需要贴出的邮资
  
/\*
  
 temp 包含邮票面值的向量。
  
 cur 当前考虑的邮票面值的索引。
  
 re 剩余需要组成的总和。
  
 m 可以使用的最大邮票数量。
  
 num 当前使用的邮票数量。
  
 return 如果可以使用给定条件组成和为're'的总和，则返回true，否则返回false。
  
 \*/
  
bool checkAnswer(vector<int>& temp, int cur, int re, int m, int num) {
  
 // 递归出口
  
 if (re == 0) return true;
  
 if (re < 0) return false;
  
 for (int i = cur; i >= 1; i--) {
  
 if (num < m) {
  
 num++;
  
 if (checkAnswer(temp, cur, re - temp[i], m, num)) // 使用了第i个面值的邮票
  
 return true;
  
 else
  
 num--; // 恢复现场
  
 }
  
 }
  
 return false;
  
}
  
/\*
  
temp:包含邮票面值的数组。
  
cur:当前考虑的邮票面值的索引。
  
m:可以使用的最大邮票数量。
  
maxt:当前已经贴出的最大邮资。
  
\*/
  
//找出由已选的前cur个面值中选,在张数不超过m的情况下能贴出的最大邮资
  
int maxl(vector<int>& temp, int cur, int m, int maxt) {
  
 int k;
  
 for (k = maxt + 1; k < m \* temp[cur]; k++) {//从maxt+1开始尝试
  
 if (!checkAnswer(temp, cur, k, m, 0))//如果不能贴出k邮资
  
 break;
  
 }
  
 return k - 1;//返回k-1
  
}
  
//递归找出最大邮资
  
/\*
  
temp:包含邮票面值的数组。
  
cur:当前考虑的邮票面值的索引。
  
m:可以使用的最大邮票数量。
  
maxt:当前已经贴出的最大邮资。
  
\*/
  
void traceback(vector<int>& temp, int cur, int maxt, int m) {
  
 if (cur == n) { //遍历到叶子
  
 if (maxRe < maxt) {
  
 maxRe = maxt;
  
 result = temp;
  
 }
  
 return;
  
 }
  
 for (int k = temp[cur] + 1; k <= maxt + 1; k++) {
  
 temp[cur + 1] = k;
  
 int max1 = maxl(temp, cur + 1, m, maxt);//下一个面值取k时能达到的最大邮资
  
 if (max1 > maxt) {//k可取
  
 traceback(temp, cur + 1, max1, m);//继续递归确定接下来的面值
  
 }
  
 }
  
}
  
  
int main() {
  
 cout << "请输入面值数量(n)：" << endl;
  
 cin >> n;
  
 cout << "请输入每张信封最多允许贴的邮票数量(m)：" << endl;
  
 cin >> m;
  
 temp.resize(n + 1);
  
 result.resize(n + 1);
  
 temp[1] = 1;
  
 traceback(temp, 1, m, m);
  
 cout << endl << "邮票面值分别为：";
  
 for (int i = 1; i <= n; i++)
  
 cout << result[i] << " ";
  
 cout << endl;
  
 cout << endl << "最大邮资区间为：1-" << maxRe << endl;
  
 return 0;
  
}