```
  while (s[start+wlen+1]<>' ') and (start+wlen<=len) do inc(wlen);
  ss:=copy(s,start+1,wlen);
  for i:=length(ss) div 2 downto 1 do
    swap(ss[i],ss[length(ss)-i+1]);
  write(ss);
  if start+wlen<=len then write(' ') else writeln;
  until start+wlen>len;
end.
```

Задача D2. ПЕРДЕ

Марийка си купила ново перде за вкъщи. За да изглежда добре окачено, тя иска кукичките, на които се закачва пердето, да са на равни разстояния една от друга. В горния край на пердето има зашити халки на равни разстояния, за които могат да се закачват кукичките. Разбира се, за да не виси пердето от двата края, първата кукичка трябва да се закачи за първата халка, а последната кукичка - за последната халка. Нека номерираме халките с целите числа и най-лявата да получи номер едно.

Помогнете на Марийка, като напишете програма **PERDE**, която, по зададен брой на халките и брой на кукичките, определя номерата на тези халки, за които са закачени кукички.

Вход: От стандартния вход се въвеждат две цели положителни числа - **A** и **B**, където **A** е броя на халките, а **B** - броя на кукичките (1<**A**<1000, 1<**B**<**A**).

Изход: На стандартния изход се извежда на екрана нарастваща редица от номерата на тези халки, за които са закачени кукички.

Тъй като не винаги е възможно всички кукички да са закачени на халки, които се намират на равни разстояния една от друга, то е необходимо поне разликата между най-голямото и най-малкото разстояние между две съседни халки да бъде минимална.

ПРИМЕР:

Вход:	**Изход:**
10 4	1 4 7 10
11 4	1 5 8 11

Решение:

Задачата се свежда до намиране на частното и остатъка от делението на A на B.

```
#include <stdio.h>

int main(int argc, char **argv) {
   int n, k;
   scanf("%d %d", &n, &k);
   printf("n=%d k=%d\n", n, k);

   int m = n - k;
   int d = k - 1;
   int l = m % d;
   int s = m / d;
   printf("m=%d d=%d l=%d s=%d\n", m, d, l, s);

   int i = 0;
```

```
    int j = 1;
    for (; i <= d; i++)
    {
        printf("%d ", j);
        j += s + 1;
        if (i >= d - l)
        {
           j++;
        }
    }
    printf("\n");

    return 0;
}
```

Задача D3. MAX3

Зададени са **N** цели числа. Напишете програма **MAX3**, която избира три от тях, произведението на които е максимално.

Вход: Данните се четат от стандартния вход. На първия ред е разположено числото **N** – брой на числата от последователността ($3 \leq \mathbf{N} \leq 10^6$). На следващите редове са разположени и самите числа, които по абсолютна стойност не надхвърлят 30 000.

Изход: На стандартния изход се извеждат три числа, произведението на които е най-голямо. Числата се извеждат в произволен ред. Ако съществуват няколко различни тройки числа, които дават максимално произведение, се извежда коя да е от тях.

ПРИМЕР:

Вход:	**Изход:**
9 3 5 1 7 9 0 9 -3 10	9 10 9
3 -5 -30000 -12	-5 -30000 -12

Решение:

Ясно е, че поради големия брой на числата, тривиалният подход за сравняване на произведенията на всички възможни тройки числа не е добро решение, защото е много бавно.

Затова е необходимо да се приложи по-добър алгоритъм. Ако последователността съдържа само неотрицателни числа, то е ясно, че максималното произведение ще се получи от трите най-големи числа в нея. Ще ги означим с max1, max2 и max3.

Но в нашия случай последователността може да съдържа и отрицателни числа, което изисква по-подробен анализ. Произведението на две отрицателни числа е положително и затова е необходимо да намерим двете най-малки отрицателни числа - min1 и min2. Тогава максимално произведение ще получим като умножим max1 с по-голямото от max2*max3 или min1*min2. Тъй като произведението на три цели числа може да стане много голямо, за да не се получи препълване (самото произведение в задачата не се търси), удачно е да се сравняват най-напред max2*max3 и min1*min2.

Ако последователността се състои само от отрицателни числа, то максималното произведение (отрицателно) ще се получи отново от max1, max2 и max3.