# **Упражнения:** *КОМБИНАТОРНИ АЛГОРИТМИ*

## **Зад. 1** *Пермутации*

*Изчислете броя на пермутациите за дадена стойност на N. Известно е, че броят на пермутациите е равен на факториела на N, т.е. N! = 1.2.3...N. Рекурсивната дефиниция е N!=N\*(N-1).*

### **Пример**

*Функцията има ограничен обхват. Тя ще работи за числа N по-малки или равни на 12. За големи стойности на N се изисква използването на дълги числа (умножение на дълги числа).*

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |
| 7 | 5040  Последното число, което се побира в int |
| 8 | 40320 |
| 9 | 362880 |
| 10 | 3628800 |
| 11 | 39916800 |
| 12 | 479001600  Последното число, което се побира в long int |
| 13 | 6227020800 |
| ... | ... |
| 30 | 265252859812191058636308480000000 |
| ... | ... |

### **Подсказки**

*Fac (k)*

*Begin*

*r:=1;*

*For i:=l To к Do r:=r\*i;*

*Faс:=r;*

*End;*

## **Зад. 2** *Сума нула*

*Нека са дадени числата a1, a2, ..., an. Да се поставят операции "+" и "–" между числата ai и ai+1, за i = 1,2, …, n–1 така, че резултатът след пресмятане на получения израз да бъде равен на 0.*

*Например, ако са дадени естествените числа от 1 до 8, то няколко възможни решения на задачата са:*

*1 + 2 + 3 + 4 – 5 – 6 – 7 + 8 = 0*

*1 + 2 + 3 – 4 + 5 – 6 + 7 – 8 = 0*

*1 + 2 – 3 + 4 + 5 + 6 – 7 – 8 = 0*

*1 + 2 – 3 – 4 – 5 – 6 + 7 + 8 = 0*

*От клавиатурата се въвежда цяло число n - броя на числата и след него n на брой числа.*

**Пример**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 8  1 2 3 4 5 6 7 8 | +1 +2 +3 +4 -5 -6 -7 +8 = 0  +1 +2 +3 -4 +5 -6 +7 -8 = 0  +1 +2 -3 +4 +5 +6 -7 -8 = 0  +1 +2 -3 -4 -5 -6 +7 +8 = 0  +1 -2 +3 -4 -5 +6 -7 +8 = 0  +1 -2 -3 +4 +5 -6 -7 +8 = 0  +1 -2 -3 +4 -5 +6 +7 -8 = 0  -1 +2 +3 -4 +5 -6 -7 +8 = 0  -1 +2 +3 -4 -5 +6 +7 -8 = 0  -1 +2 -3 +4 +5 -6 +7 -8 = 0 |

### **Подсказки**

*Генерирайте всевъзможните вариации с повторение на n–1 елемента от втори клас, т.е. всевъзможните наредени (n–1)-орки, съставени от 0 и 1 (което отговаря на положителен и отрицателен знак пред съответното число). За всяка такава (n–1)-орка проверете дали е решение на задачата, като за целта пресметнете стойността на съответния израз.*

## **Зад. 3** *Разбиване на число (като сума от числа)*

*По дадено естествено число n да се намерят всички възможни не наредени представяния (разбивания) на n като сума от естествени числа (не непременно различни). Така например, числото 5 може да се разбие по следните 7 начина:*

*5 = 5*

*5 = 4 + 1*

*5 = 3 + 2*

*5 = 3 + 1 + 1*

*5 = 2 + 2 + 1*

*5 = 2 + 1 + 1 + 1*

*5 = 1 + 1 + 1 + 1 + 1*

*От клавиатурата се въвежда едно цяло число n - числото за“разбиване”.*

**Пример**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 7 | 6+1  5+2  5+1+1  4+3  4+2+1  4+1+1+1  3+3+1  3+2+2  3+2+1+1  3+1+1+1+1  2+2+2+1  2+2+1+1+1  2+1+1+1+1+1  1+1+1+1+1+1+1 |

### 

### **Подсказки**

*Може да използвате рекурсивен алгоритъм:*

*разбиване(0) = {}*

*разбиване(n) = {k} + разбиване(n–k), където k = n, n–1,…,1.*

*Трябва да внимавате и да избегнете генериране на повтарящи се разбивания, като:*

*5 = 3 + 2*

*5 = 2 + 3*

*Всяко следващо събираемо е необходимо да бъде по-малко или равно на предходното. Рекурсивната функция, извършваща разбиването, има два аргумента: n (число за разбиване) и променлива, показваща колко пъти досега е било разбивано числото.*

## **Зад. 4** *Разбиване на число (като произведение от числа)*

*По дадено естествено число n да се намерят всички възможни не наредени представяния (разбивания) на n като произведение от естествени числа (не непременно различни).*

*От клавиатурата се въвежда едно цяло число n.*

**Пример**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 50 | 25 \* 2  10 \* 5  5 \* 5 \* 2 |

### 

### **Подсказки**

*Алгоритъмът, по който можете да реализирате разлагането, е аналогичен на този, който е описан в задача 3. Вместо devNum(n-k,cnt+1) ще извикваме рекурсивно devNum(n/k,cnt+1), при това не за всяко k, а само за тези, за които n % k == 0. Условието за продължаване на разбиването (цикъла for) ще бъде k > 1, а не k ≥ 1, т.е. дъното на рекурсията ще бъде k == 1, а не k == 0 (последното се обяснява лесно: 0 и 1 са именно идентитетите на операциите събиране и умножение).*

## **Зад. 4** *Разбиване на числа (като сума от дадени числа)*

*Нека са дадени пощенски марки от 2, 5 и 10 лева . Трябва да се изпратим колет на стойност 20 лева. Всички възможности (общо 6 на брой) за образуване на тази сума са:*

*20 = 10 + 10*

*20 = 10 + 5 + 5*

*20 = 10 + 2 + 2 + 2 + 2 + 2*

*20 = 5 + 5 + 5 + 5*

*20 = 5 + 5 + 2 + 2 + 2 + 2 + 2*

*20 = 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2*

*От клавиатурата последователно се въвеждат числата n - стойността на колета, m - броя на наличните марки, и m на брой числа - стойностите на марките. Всички числа са цели числа.*

**Пример**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 20 3  2 5 10 | 5 + 5 + 5  5 + 5 + 3 + 2  5 + 3 + 3 + 2 + 2  5 + 2 + 2 + 2 + 2 + 2  3 + 3 + 3 + 3 + 3  3 + 3 + 3 + 2 + 2 + 2  3 + 2 + 2 + 2 + 2 + 2 + 2 |

### 

### **Подсказки**

*Алгоритъмът за решаването на тази задача е подобен на този за разбиване на число като сума от естествени числа. Пазете числата, които можете да ползвате при разбиването, в масив given[gN]. Цикълът ще се изпълнява за p = 0,1,...,gN-1, като при рекурсивното извикване вместо с p, намалете n със съответната стойност на given[p].*