# Подготовка за изпит по "Програмиране за начинаещи"

Задачи за тренировка за междинен изпит по [„Основи на програмирането“ @ СофтУни](https://softuni.bg/courses/programming-basics) (2 март 2016).

## Лице на триъгълник в равнината

**Триъгълник в равнината** е зададен чрез координатите на трите си върха. Първо е зададен **върхът** (**x1**, **y1**). След това са зададени останалите два върха: (**x2**, **y2**) и (**x3**, **y3**), които **лежат на обща хоризонтална права** (т.е. имат еднакви **Y** координати). Напишете програма, която пресмята **лицето на триъгълника** по координатите на трите му върха.

### Вход

От конзолата се четат **6 цели числа** (по едно на ред):  
x1, y1, x2, y2, x3, y3.

* Всички входни числа са в диапазона [**-1000**…**1000**].
* Гарантирано е, че y2 = y3.

### Изход

**Лицето на триъгълника** да се отпечата на конзолата.

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Чертеж** | **Обяснения** |
| 5  -2  6  1  1  1 | 7.5 |  | Страната на триъгълника **a** = 6-1 = **5**.  Височината на триъгълника **h** = 1 - (-2) = **3**.  Лицето на триъгълника **S** = a \* h / 2 = 5 \* 3 / 2 = **7.5**. |
| 4  1  -1  -3  3  -3 | 8 |  | Страната на триъгълника **a** = 3-(-1) = **4**.  Височината на триъгълника **h** = 1 - (-3) = **4**.  Лицето на триъгълника **S** = a \* h / 2 = 4 \* 4 / 2 = **8**. |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#0>.

## Пренасяне на тухли

Строителни работници трябва да пренесат общо x **тухли**. **Работниците** са w на брой и работят едновременно. Те превозват тухлите в колички, всяка с **вместимост** m тухли. Напишете програма, която прочита целите числа x, w и m и пресмята **колко най-малко курса** трябва да направят работниците за да превозят тухлите.

### Вход

От конзолата се четат **3 цели числа** (по едно на ред):

* **Броят** **тухли** x се чете от първия ред.
* **Броят** **работници** w се чете от втория ред.
* **Вместимостта** **на количката** m се чете от третия ред.

Всички входни числа са цели и в диапазона [**1**…**1000**].

### Изход

Да се отпечата на конзолата **минималният брой курсове**, необходими за превозване на тухлите.

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 120  2  30 | 2 | Имаме **2** работника, всеки вози по **30** тухли на курс. Общо работниците возят по **60** тухли на курс. За да превозят **120** тухли, са необходими точно **2 курса**. |
| 355  3  10 | 12 | Имаме **3** работника, всеки вози по **10** тухли на курс. Общо работниците возят по **30** тухли на курс. За да превозят **310** тухли, са необходими **12** курса: **11** пълни курса превозват **330** тухли и последният **12**-ти курс пренася последните **25** тухли. |
| 5  12  30 | 1 | Имаме **5** работника, всеки вози по **30** тухли на курс. Общо работниците возят по **150** тухли на курс. За да превозят **5** тухли, е достатъчен само **1** курс (макар и непълен, само с **5** тухли). |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#1>.

## Точка върху отсечка

Върху хоризонтална права е разположена **хоризонтална отсечка**, зададенас x координатите на двата си края: first и second. Точка е разположена върху същата хоризонтална права и е зададена с x координатата си. Напишете програма, проверява дали точката е **вътре или вън от отсечката** и изчислява **разстоянието до по-близкия край** на отсечката.

### Вход

От конзолата се четат **3 цели числа** (по едно на ред):

* На първия ред стои числотоfirst – единият край на отсечката.
* На втория ред стои числотоsecond – другият край на отсечката.
* На третия ред стои числотоpoint – местоположението на точката.

Всички входни числа са цели и в диапазона [**-1000**…**1000**].

### Изход

Резултатът да се отпечата на конзолата:

* На първия ред да се отпечата “in” или “out” – дали точката е върху отсечката или извън нея.
* На втория ред да се отпечата разстоянието от точката от най-близкия край на отсечката.

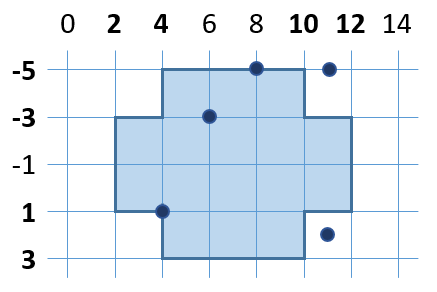
### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Визуализация** |
| 10  5  7 | in  2 |  |
| 8  10  5 | out  3 |  |
| 1  -2  3 | out  2 |  |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#2>.

## Точка във фигурата

Да се напише програма, която проверява дали дадена точка (x, y) е **вътре** или **извън** следната фигура:



### Вход

От конзолата се четат **2 цели числа** (по едно на ред): x и y.

Всички входни числа са цели и в диапазона [**-1000**…**1000**].

### Изход

Да се отпечата на конзолата “in” или “out” – дали точката е **вътре** или **извън** фигурата (на контура е вътре).

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 8  -5 | in | 6  -3 | in | 11  -5 | out | 11  2 | out |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#3>.

## Дата след 5 дни

Дадени са две числа d (ден) и m (месец), които формират **дата**. Да се напише програма, която отпечатва датата, която ще дойде **след 5 дни**. Например 5 дни след **28.03** е датата **2.04**. Приемаме, че месеци април, юни, септември и ноември имат по 30 дни, февруари има 28 дни, а останалите месеци имат по 31 дни. Месеците да се отпечатат с **водеща нула**, когато са едноцифрени (например 01, 08, 12).

### Вход

Входът се чете от конзолата и се състои от два реда:

* На първия ред стои едно цяло число d в интервала [1…31] – ден. Номерът на деня не надвишава броя дни в съответния месец (например 28 за февруари).
* На втория ред стои едно цяло число m в интервала [1…12] – месец. Месец 1 е януари, месец 2 е февруари, …, месец 12 е декември. Месецът може да съдържа водеща нула (например април може да бъде изписан като 4 или 04).

### Изход

Отпечатайте на конзолата един единствен ред, съдържащ дата след 5 дни във формат ден.месец. Месецът трябва да бъде двуцифрено число, с водеща нула ако е необходимо. Денят трябва да е без водеща нула.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 28  03 | 2.04 | 27  12 | 1.01 | 25  1 | 30.01 | 26  02 | 3.03 |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#4>.

## Суми от 3 числа

Дадени са **3 цели числа**. Да се напише програма, която проверява дали **сумата на две от числата е равна на третото**. Например, ако числата са **3, 5 и 2**, сумата на две от числата е равна на третото: **2 + 3 = 5**.

### Вход

От конзолата се четат **три цели числа**, по едно на ред. Числата са в диапазона [1…1000].

### Изход

* Да се отпечата на конзолата един ред, съдържащ решението на задачата във формат “A + B = C”, където A, B и C са измежду входните три числа и A ≤ B.
* Ако задачата няма решение, да се отпечата „No“ на конзолата.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3  5  2 | 2 + 3 = 5 | 2  2  4 | 2 + 2 = 4 | 1  1  5 | No | 2  6  3 | No |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#5>.

## Суми през 3

Дадени са **n** цели числа **a1**, **a2**, …, **an**. Да се пресметнат сумите:

* **sum1** = **a1** + **a4** + **a7** + **…** (сумират се числата започвайки от първото със стъпка 3)
* **sum2** = **a2** + **a5** + **a8** + **…** (сумират се числата започвайки от второто със стъпка 3)
* **sum3** = **a3** + **a6** + **a9** + **…** (сумират се числата започвайки от третото със стъпка 3)

### Вход

Входните данни се четат от конзолата. На първия ред стои цяло число **n** (**0** ≤ **n** ≤ **1000**). На следващите **n** реда стоят **n** цели числа в интервала [**-1000**…**1000**]: **a1**, **a2**, …, **an**.

### Изход

На конзолата трябва да се отпечатат 3 реда, съдържащи търсените 3 суми, във формат като в примерите.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| **2**  **3**  **5** | **sum1 = 3**  **sum2 = 5**  **sum3 = 0** | **4**  **7**  **-2**  **6**  **12** | **sum1 = 19**  **sum2 = -2**  **sum3 = 6** | **5**  **3**  **5**  **2**  **7**  **8** | **sum1 = 10**  **sum2 = 13**  **sum3 = 2** |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#6>.

## Поредица от нарастващи елементи

Дадена е редица от **n** числа: **a1**, **a2**, …, **an**. Да се пресметне **дължината на най-дългата нарастваща поредица** от последователни елементи в редицата от числа.

### Вход

Входните данни се четат от конзолата. На първия ред стои цяло число **n** (**0** ≤ **n** ≤ **1000**). На следващите **n** реда стоят **n** цели числа в интервала [**-1000**…**1000**]: **a1**, **a2**, …, **an**.

### Изход

На конзолата трябва да се отпечата едно число – **дължината** на най-дългата нарастваща редица.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3  5  **2**  **4** | 2 | 4  **2**  **8**  7  6 | 2 | 4  **1**  **2**  **4**  4 | 3 | 4  **5**  **6**  **7**  **8** | 4 |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#7>.

## Перфектен диамант

Да се напише програма, която прочита от конзолата цяло число n и чертае **перфектен** **диамант** с размер n като в примерите по-долу.

### Вход

Входът е цяло число n в интервала [**1**…**1000**].

### Изход

На конзолата трябва да се отпечата диамантът като в примерите.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | \*  \*-\*  \* | 3 | \*  \*-\*  \*-\*-\*  \*-\*  \* | 4 | \*  \*-\*  \*-\*-\*  \*-\*-\*-\*  \*-\*-\*  \*-\*  \* | 5 | \*  \*-\*  \*-\*-\*  \*-\*-\*-\*  \*-\*-\*-\*-\*  \*-\*-\*-\*  \*-\*-\*  \*-\*  \* |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#8>.

## Правоъгълник със звездички в центъра

Да се напише програма, която прочита от конзолата цяло число n и чертае **правоъгълник** с размер n **с две звездички в центъра**, като в примерите по-долу.

### Вход

Входът е цяло число n в интервала [**2**…**1000**].

### Изход

На конзолата трябва да се отпечата диамантът като в примерите.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | %%%%  %\*\*%  %%%% | 3 | %%%%%%  % %  % \*\* %  % %  %%%%%% | 4 | %%%%%%%%  % %  % \*\* %  % %  %%%%%%%% | 5 | %%%%%%%%%%  % %  % %  % \*\* %  % %  % %  %%%%%%%%%% |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#9>.

## Четворки нарастващи числа

По дадена двойка числа **a** и **b** да се генерират всички четворки **n1**, **n2**, **n3**, **n4**, за които **a** ≤ **n1** < **n2** < **n3** < **n4** ≤ **b**.

### Вход

Входът съдържа две цели числа a и b в интервала [b…**1000**], по едно на ред.

### Изход

Изходът съдържа всички търсени **четворки числа**, в нарастващ ред, по една на ред.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3  7 | 3 4 5 6  3 4 5 7  3 4 6 7  3 5 6 7  4 5 6 7 | 15  20 | 15 16 17 18  15 16 17 19  15 16 17 20  15 16 18 19  15 16 18 20  15 16 19 20  15 17 18 19  15 17 18 20  15 17 19 20  15 18 19 20  16 17 18 19  16 17 18 20  16 17 19 20  16 18 19 20  17 18 19 20 | 5  7 | No | 10  13 | 10 11 12 13 |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#10>.

## Генериране на правоъгълници

По дадено число n и **минимална площ** m да се генерират всички правоъгълници с цели координати в интервала [-n…n] с площ поне m. Генерираните правоъгълници да се отпечатат в следния формат:

|  |
| --- |
| (left, top) (right, bottom) -> area |

Правоъгълниците се задават чрез горния си ляв и долния си десен ъгъл. В сила са следните неравенства:

* -n ≤ left < right ≤ n
* -n ≤ top < bottom ≤ n

### Вход

От конзолата се въвеждат две числа, по едно на ред:

* Цяло число n в интервала [**1**…**100**] – задава минималната и максималната координата на връх.
* Цяло число m в интервала [**0**…**50000**] – задава минималната площ на генерираните правоъгълници.

### Изход

* На конзолата трябва да се отпечатат описаните правоъгълници във формат като в примерите по-долу.
* Ако за числата n и m няма нито един правоъгълник, да се изведе “No”.
* Редът на извеждане на правоъгълниците е без значение.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1  2 | (-1, -1) (0, 1) -> 2  (-1, -1) (1, 0) -> 2  (-1, -1) (1, 1) -> 4  (-1, 0) (1, 1) -> 2  (0, -1) (1, 1) -> 2 | 2  17 | No | 3  36 | (-3, -3) (3, 3) -> 36 |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/157#11>.