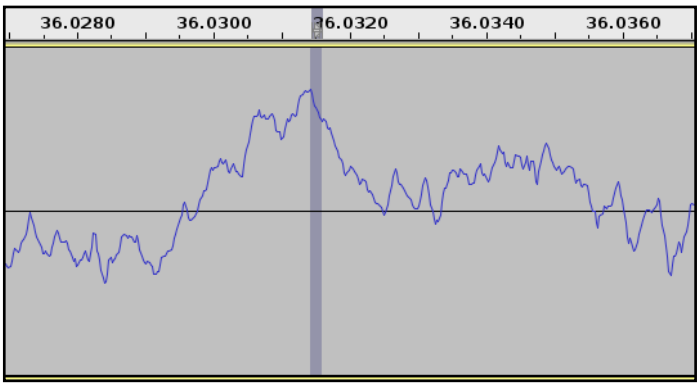
НАЦИОНАЛЕН ПРОЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Пловдив, 10 – 12 юни 2016 г.
Група В, 9 – 10 клас

Задача В1. АУДИО КОМПРЕСИЯ



Суровият вид на цифровото аудио обикновено е серия числа (тук ще приемем, че са 16-битови), които кодират моментната амплитуда на звуковите вълни, с десетки хиляди отчета в секунда. В картинката горе се вижда визуализация на част от звук, а данните за порцията в тъмната ивица може да изглеждат така:
21581, 21338, 21073, 20815, 20555, 20282, …

Кодирането на тези числа изисква по 16 бита на отчет (поддържа се обхватът от – 32768 до + 32767), но лесно се вижда, че реално стойностите не се изменят много драстично и една техника за компресирането на тези данни без загуба, се нарича *delta encoding*. Вместо самите стойности, ще записваме разликите (наречени „***делти***“) между последователните отчети, като ще си представим, че в началото сме поставили фиктивен отчет със стойност 0:
+ 21581, – 243, – 265, – 258, – 260, – 273, …

Като изключим началната стойност, всички останали *делти* могат да бъдат описани само в 10 бита, вместо в 16 (10-битовите числа със знак имат обхват от – 512 до + 511). Примерният компресиран *bitstream* може да изглежда така:

№	Тип	Стойност	Битов код	Дължина
1	Делта стойност	+ 21581	0101 0100 0100 1101	16
2	Смяна на ширината	→10 бита	1000 0000 0000 0000 <u>1001</u>	20
3	Делта стойност	– 243	10 1111 0011	10
4	Делта стойност	– 265	11 0000 1001	10
5	Делта стойност	– 258	11 0000 0010	10
6	Делта стойност	– 260	11 0000 0100	10
7	Делта стойност	– 273	11 0001 0001	10
			Общо:	**86**

Така трансформираните данни заемат само 86 бита, вместо 6×16 = 96. На практика, за по-дълги файлове компресията е значително по-голяма.

Специалният код за смяна на ширината (СШ) се образува от кода на минималното число за текущата ширина (в случая $1000\ 0000\ 0000\ 0000_2 = -32768_{10}$), следвано от 4-битов код за новата ширина (<u>подчертан</u> в примера). Тя може да бъде между 1 и 17 бита, по хитрата схема за „прескачане“ на текущата ширина, например:

№	Текуща ширина	Код	Нова ширина
1	8	0000	1
2	8	0001	2
3	8	0110	7
4	8	0111	9
5	8	1111	17
6	10	0111	8

Идеята тук е, че ако в момента сме със ширина 8 бита, няма смисъл да излъчим смяна към … отново 8 бита. Като премахнем („прескочим“) този вариант, 16-те възможни 4-битови кода могат да укажат всяко число от 1 до 17, без текущото. Забележете как едни и същи кодове (в редове 4 и 6) означават различни нови ширини, в зависимост от контекста.

НАЦИОНАЛЕН ПРОЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Пловдив, 10 – 12 юни 2016 г.
Група В, 9 – 10 клас

Може би се чудите за какво са ни 17-битовите числа. Възможно е делтата между две 16-битови числа да не може да се запише в 16 бита. Освен това, ако първият отчет във файла беше – 32768 (валидно 16-битово число), трябва да го кодираме с делта -32768, което пък съвпада с 16-битовия маркер за СШ. Затова трябва да поддържаме 17-битови кодове и делти (които на практика се налагат много рядко). Освен това, в действителност компресията на файла започва с ширина 17 бита (по подразбиране) с цел да се избегнат проблеми като гореописаните. В такъв смисъл, примерният битстрийм е леко грешен – реално позиции 1 и 2 изискват, респективно, 17 и 21 бита и цената е 88 бита общо (в тестове по-долу това наистина е така).

В резюме, *delta encoding* компресирането изглежда така:

1. Започваме с предходен отчет (фиктивен) = 0 и ширина $W = 17$.
2. За всеки отчет във файла емитираме делтата спрямо предходния, като разрешеният обхват делти е в интервала $[-(2^{W-1}-1), +(2^{W-1}-1)]$. Цената на една делта е W бита.
3. Смяната на ширината ($W_{old} \rightarrow W_{new}$) е разрешена по всяко време и коства $W_{old} + 4$ бита.

Разбира се, разположението на маркерите за СШ е ключово. В примера горе наблюдателните от вас сигурно са забелязали, че делтата – 243 можеше да се кодира само в 9 бита. Битстриймът можеше да бъде [+ 21581, СШ→9, – 243, СШ→10, –265, …] – но тук цената на допълнителната СШ напълно би заличила спестения бит. Целта на задачата **`impulse`** е да намерите оптимално разположение на СШ – трябва да компресирате даден входен файл с минимален брой битове, като от вас се изисква да изчислите само броя им – самият битстрийм или позициите на СШ не са необходими.

Вход

От стандартния вход се въвежда един ред с числото N – броя на отчетите във входа. Следват N реда с по едно цяло число, всяко в интервала [– 32768, + 32767], задаващи самите отчети.

Изход

Програмата трябва да извежда на стандартния изход едно единствено число – минималния брой битове, който е достатъчен за кодиране отчетите, зададени на входа, чрез описания алгоритъм *data encoding*.

Примери

Вход 1	Изход 1	Вход 2	Изход 2
6 21581 21338 21073 20815 20555 20282	88	9 42 42 42 42 42 42 48 32767 -32768	94
***Пояснение**: Това е примерът от условието, с корекцията за 17-битовите числа в началото*		***Пояснение**: Битстриймът е* [*+42, СШ→1, +0, +0, +0, +0, +0, СШ→4, +6, СШ→17, +32719, -65535*]. *Цената на първата СШ е 21 бита, втората коства 5 бита, а третата – 8.*	

НАЦИОНАЛЕН ПРОЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Пловдив, 10 – 12 юни 2016 г.
Група В, 9 – 10 клас

Ограничения

$1 \leq N \leq 10^6$

В поне 50% от тестовете N няма да надвишава 1000.

Ограничение за използваната памет – 1 MB.

Забележка

В примерите е използван стандартният т. нар. „допълнителен код“ за двоично кодиране на отрицателни числа. Тъй като в задачата не е необходимо да определяте самия битстрийм-изход, конкретните двоични кодирания на отрицателните стойности са дадени само за изясняване на проблема и задоволяване на любопитството ви и нямат отношение към крайното решение.

НАЦИОНАЛЕН ПРОЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Пловдив, 10 – 12 юни 2016 г.
Група В, 9 – 10 клас

Задача В2. ПРОИЗВЕДЕНИЕ

Дадена е редица от различни цели положителни числа a_1, a_2, …, a_n. Напишете програма **`prod`**, която намира 3 елемента a_i, a_j, a_k от дадената редица, такива че $i < j < k$, $a_i < a_j < a_k$ и произведението $a_i \cdot a_j \cdot a_k$ да има максимална стойност.

Вход

Числата от дадената редица, разделени с интервали.

Изход

На първия ред на изхода да бъде изведената стойността на максималното произведение. На следващия ред да се изведат три цели числа (разделени с по един интервал), равни на намерените индекси $i < j < k$, за които се получава максималното произведение. Ако не съществуват такива индекси, програмата трябва да изведе само числото 0. Ако съществува повече от една такава тройка индекси, програмата трябва да изведе коя да е от тях.

Ограничения

Броят на елементите на дадената редица не е по-голям от 200 000.

Всеки елемент на редицата е цяло положително число, което не е по-голямо от 1 000 000.

Пример

Вход

```
7 2 10 8 3 6 9 12 4 11
```

Изход

```
864
4 7 8
```

НАЦИОНАЛЕН ПРОЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Пловдив, 10 – 12 юни 2016 г.
Група В, 9 – 10 клас

Задача В3. ИГРА С ЧИСЛО

Цялото положително число *A* се записва само с единици и двойки. Докато скучаеше, Пенчо измисли простичка игра, с която да си запълни времето. За един ход той:

- избира една от цифрите на *A*, преброява колко единици има преди нея и ако те са нечетен брой променя избраната цифра (от единица на двойка или от двойка на единица), а ако са четен брой – не променя избраната цифра;
- премества първата цифра на числото в края му.

С новополученото число Пенчо извършва същите действия. След като направи *Q* хода, той започна да се съмнява дали не се е объркал някъде при преписването или при броенето и затова ви моли да напишете програма **bnum**, с която да си проверява крайните резултати.

Вход

На първия ред на стандартния вход е зададено числото *A*. На следващия ред е зададен броят на ходовете *Q*. Следват *Q* реда, всеки от които съдържа по едно цяло число – избраната от Пенчо позиция за поредния ход. Позициите на числата са номерирани отляво надясно, като най-лявата позиция има номер 1.

Изход

На един ред на стандартния изход програмата трябва да изведе числото, което Пенчо трябва да получи след описаните *Q* хода.

Ограничения

Числото *A* съдържа поне две и най-много 1 000 000 цифри. Пенчо извършва поне един и най-много 1 000 000 хода.

В 20% от тестовите примери *A* има най-много 1 000 цифри и $Q < 1\,000$.

Пример

Вход

```
2211211111
5
2
8
7
5
3
```

Изход

```
1221122112
```

Пояснение на примера

Позиция	1	2	3	4	5	6	7	8	9	10
Начално число	**2**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**
Ход 1	2	2	1	1	2	1	1	1	1	1
	2	1	1	2	1	1	1	1	1	2
Ход 2	2	1	1	2	1	1	1	2	1	2
	1	1	2	1	1	1	2	1	2	2
Ход 3	1	1	2	1	1	1	1	1	2	2
	1	2	1	1	1	1	1	2	2	1
Ход 4	1	2	1	1	2	1	1	2	2	1
	2	1	1	2	1	1	2	2	1	1
Ход 5	2	1	2	2	1	1	2	2	1	1
	1	**2**	**2**	**1**	**1**	**2**	**2**	**1**	**1**	**2**