Допълнителни задачи за списъци

Практически упражнения към курса **"Programming Fundamentals" за ученици**.
Тествайте задачите от тази тема в judge: https://judge.softuni.bg/Contests/2661

1. Сума на съседни еднакви числа

Напишете програма, която **сумира всички съседни еднакви числа** в списък от цели числа, започвайки **отляво надясно**.

- След като две числа са сумирани, полученият резултат може да бъде равен на някой от другите му съседи, което означава, че също трябва да се сумира (вижте примерите).
- Винаги сумирайте **най-левите** две еднакви числа (ако има няколко двойки от еднакви числа).

Примери

Вход	Изход	Обяснение
3 3 6 1	12 1	**3 3** 6 1 → **6 6** 1 → 12 1
8 2 2 4 8 16	16 8 16	8 **2 2** 4 8 16 → 8 **4 4** 8 16 → **8 8** 8 16 → 16 8 16
5 4 2 1 1 4	5 8 4	5 4 2 **1 1** 4 → 5 4 **2 2** 4 → 5 **4 4** 4 → 5 8 4

Подсказки

1. Въведете числата и създайте **списък от числа**.
2. Намерете двете **най-леви съседни еднакви клетки**.
3. **Заменете** ги с тяхната **сума**.
4. **Повторете** (1) и (2) докато не остават две съседни еднакви клетки.
5. **Изведете** обработения списък.

2. Отделяне по регистър на дума

Въведете **text**, след което го разделете към думи и ги разпредете в **3 списъка**.

- **Думи с малки букви** като “programming”, “at” и“databases” – съдържащи се само от малки букви.
- **Думи с големи букви** като “PHP”, “JS” and “SQL” – съдържат само големи букви.
- **Смесени думи** като “C#”, “CodeCamp” и “Java” – всички други.

Използвайте следните **разделители** между думите: **`, ; : . ! ( ) " ' \ / [ ]` интервал**

Изведете трите списъка, както е показано в примера.

Примери

Вход	Изход

`Learn programming at CodeCamp: Java, PHP, JS, HTML 5, CSS, Web, C#, SQL, databases, AJAX, etc.`	`Lower-case: programming, at, databases, etc`
	`Mixed-case: Learn, CodeCamp, Java, 5, Web, C#`
	`Upper-case: PHP, JS, HTML, CSS, SQL, AJAX`

Подсказки

- **Отделете** входния текст чрез използваните по-горе **разделители**.
- **Обработете** получения **списък от думи** една по една.

- Създайте 3 списъка от думи (в началото празни): думи с малки букви, думи с големи букви, думи със смесени букви.
- Проверете всяка дума и я разпределете към някой от трите списъка:
  - Пребройте всички **малки букви** и **големи букви**.
  - Ако всичките букви са **малки**, добавете думата към списък на думите с малки букви
  - Ако всичките букви са **големи**, добавете думата към списък на думите с големи букви
  - В противен случай се смята, че думата е със смесени букви → добавяме я към списъка на думите със смесени букви.
- Изведете получените списъци, както е показано в списъка горе.

3. Променлив списък

Напишете програма, която въвежда **списък от цели числа** от **конзолата** и получава **команди**, които **манипулират** списъка. Вашата програма може да получава следните команди:

- **`Delete {елемент}`** – изтрива всички елементи в списъка, които са равни на дадения елемент
- **`Insert {елемент} {позиция}`** – вмъква елемент на дадената позиция

Програмата трябва да приключва, когато получи команда **Odd** или **Even**. Ако програмата получи Odd ➔ извежда всички **нечетни** числа в списъка отделени с **единствен** интервал, иначе извеждаме по същия начин всички **четни** числа.

Примери

Вход	Изход
1 2 3 4 5 5 5 6 Delete 5 Insert 10 1 Delete 5 Odd	1 3

Вход	Изход
20 12 4 319 21 31234 2 41 23 4 Insert 50 2 Insert 50 5 Delete 4 Even	20 12 50 50 31234 2

4. Търсене на число

На **първия ред** се въвежда **списък от цели числа**. На **следващия** ред, ще получите **списък** с точно **три числа**. **Първото** от тях показва **броя на елементите,** които трябва да **вземете** от **списъка** (**считано от първия елемент**). **Второто** число показва **броя на елементите**, които трябва да **изтриете** от елементите, които взехте (**считано от първия елемент**). **Последното число** е това, което търсим в получения **списък** след манипулациите. Ако това число е в списъка, извеждаме: "**YES!**", в противен случай "**NO!**"

Примери

Вход	Изход
1 2 3 4 5 6 5 2 3	YES!

Вход	Изход
12 412 123 21 654 34 65 3 23 7 4 21	NO!

** Най-дълга нарастваща поредица (Longest Increasing Subsequence – LIS)

Въведете списък от цели числа и намерете **най-дългата растяща подредица** (LIS). Ако има няколко такива, изведете **най-лявата**.

Примери

Вход	Изход
1	1
7 **3 5** 8 -1 0 **6 7**	3 5 6 7
1 2 5 **3 5** 2 4 1	1 2 3 5
0 10 20 30 30 40 **1** 50 **2 3 4 5 6**	0 1 2 3 4 5 6
11 12 13 **3** 14 **4** 15 **5 6 7 8** 7 **16** 9 8	3 4 5 6 7 8 16
3 14 **5** 12 15 **7 8 9 11** 10 1	3 5 7 8 9 11

Подсказки

- Нека имаме **n** числа в списъка **`nums[0…n-1]`**.
- Нека **`len[p]`** показва дължината на най-дългата растяща поредица (LIS) завършваща в позиция **p**.
- Във for-цикъл, трябва да изчислим **`len[p]`** за **p** = **0** … **`n-1`** както следва:
  - Нека **`left`** е най-лявата позицията наляво от **p** (**`left`** < **p**), така щото **`len[left]`** да е колкото се може по-голямо.
  - Тогава, **`len[p]`** = **1** + **`len[left]`**. Ако **`left`** не съществува, **`len[p]`** = **1**.
  - Също така, запазете **`prev[p]`** = **`left`** (запазваме си в **`prev[]`** предната позицията, която сме използвали за да получим най-добрата дължина за позицията **p**).
- Веднъж щом стойностите на **`len[0`**…**`n-1]`** са изчислени, върнете най-дългата растяща подредица започвайки от **p** така че **`len[p]`** да е максималното и се върнете назад чрез **p** = **`prev[p]`**.
- Тази таблица илюстрира изчисленията за последния пример:

`index`	0	1	2	3	4	5	6	7	8	9	10
`nums[]`	**3**	**14**	**5**	**12**	**15**	**7**	**8**	**9**	**11**	**10**	**1**
`len[]`	1	2	2	3	4	3	4	5	6	6	1
`prev[]`	-1	0	0	2	3	2	5	6	7	7	-1
`LIS`	{3}	{3,14}	{3,5}	{3,5,12}	{3,5,12,15}	{3,5,7}	{3,5,7,8}	{3,5,7,8,9}	{3,5,7,8,9,11}	{3,5,7,8,9,10}	{1}

5. * Списъчен манипулатор

Напишете програма, която **въвежда списък от цели числа** от конзолата и **списък от команди**, които се **изпълняват върху списъка**. Командите са както следва:

- **add <индекс> <елемент>** – вмъква елемент на зададената позиция (елементите надясно от тази позиция включително се изместват надясно).
- **addMany <индекс> <елемент 1> <елемент 2> … <елемент n>** – добавя множество от елементи на дадената позиция.
- **contains <елемент>** – отпечатва индекса на първото срещане на зададения елемент (ако съществува) в списъка или **-1**, ако елемента не е открит.
- **remove <индекс>** – премахва елемента, намиращ се на зададената позиция
- **shift <позиции>** – **отмества всеки елемент** от списъка съответния брой позиции **наляво** (с ротация).
  - Например, [1, 2, 3, 4, 5] -> shift 2 -> [3, 4, 5, 1, 2]
- **sumPairs** – сумира елементите на всички двойки в списъка (първа + втора, трета + четвърта, …).
  - Например, [1, 2, 4, 5, 6, 7, 8] -> [3, 9, 13, 8].
- **print** – спира да получава повече команди и извежда последното състояние на списъка.

Примери

Вход	Изход
1 2 4 5 6 7 add 1 8 contains 1 contains -3 print	0 -1 [1, 8, 2, 4, 5, 6, 7]
1 2 3 4 5 addMany 5 9 8 7 6 5 contains 15 remove 3 shift 1 print	-1 [2, 3, 5, 9, 8, 7, 6, 5, 1]
2 2 4 2 4 add 1 4 sumPairs print	[6, 6, 6]
1 2 1 2 1 2 1 2 1 2 1 2 sumPairs sumPairs addMany 0 -1 -2 -3 print	[-1, -2, -3, 6, 6, 6]

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).