

## Тестовое задание на стажировку в СКБ Контур (разработчики)

В этом году вам предстоит пройти серьезное испытание – чтобы успешно решить тестовое задание, вам понадобится продемонстрировать знание основных алгоритмов и структур данных, умение писать программы на языке C#, читать документацию, разбираться в тонкостях языка и грамотно тестировать решения. Готовы ли вы к этому вызову?

### Как пройти отбор?

- Вам предлагается две задачи. За каждую вы можете получить от 1 до 5 баллов, однако, если решение первой задачи нас не удовлетворит, то вторую задачу мы даже не будем проверять. Задачи можно сдавать по отдельности.
- Среди кандидатов, набравших наибольшее общее количество баллов, отбор будет проводиться по итогам собеседований.
- Если вы пришлете решение первой задачи *заранее*, то сначала **1 апреля**, а потом **29 апреля** мы вышлем вашу текущую оценку по ней, и вы сможете попробовать улучшить ее, отправив другое решение (будет учтена лучшая из всех оценок).

Пожалуйста, отправляйте заполненные заявки, решения и любые вопросы по условиям стажировки или тестовому заданию на [kontur-student@skbkontur.ru](mailto:kontur-student@skbkontur.ru)

Желаем удачи!

### Задача 1. Pagination

Представьте, что ваш коллега-нытик пришел рассказать о своей непростой задаче – ему нужно не просто упорядочить по возрастанию набор целых чисел, а выдать все элементы упорядоченного набора с  $L$ -го по  $R$ -й включительно!

Вы заявили, что это элементарная задача и, чтобы написать решение на языке C#, вам нужно десять минут. Ну, или час. Или два. Или шоколадка «Аленка»...

Напишите консольное приложение на языке C#, которое будет читать входные данные из файла *input.txt* и выводить результат в файл *output.txt*, расположенный в текущем каталоге.

Формат входных данных устроен следующим образом. В первой строке записано целое число  $N$  – количество чисел в наборе. Во второй строке через пробел записаны  $N$  чисел – элементы этого набора. В третьей строке записано целое число  $Q$  – количество запросов, которые нужно обработать. В следующих  $Q$  строках описаны эти запросы. Если запрос имеет вид «+  $X$ », то необходимо добавить число  $X$  в набор. Если запрос имеет вид «?  $L R$ », то нужно выдать в отдельной строке элементы текущего набора, упорядоченного по неубыванию, с  $L$ -го по  $R$ -й включительно. Гарантируется, что все запросы корректны ( $1 \leq L \leq R$ ;  $R$  не превосходит количества чисел в текущем наборе).

В наборе в любой момент времени будут содержаться только целые числа типа `int`. Гарантируется, что количество чисел в наборе в любой момент времени и общее количество чисел, которое нужно вывести для всех запросов, не будет превосходить  $10^6$ .

## Пример

Входные данные (input.txt)	Результат (output.txt)
5	2 3 4
5 4 3 2 1	1 2 3
5	-3 1 1 2 3 4 5
? 2 4	
+ -3	
? 2 4	
+ 1	
? 1 7	

Ваш код будут оценивать и тестировать три программиста:

- Билл будет запускать ваше решение на тестах размером не больше 10Кб.
- В тестах Стивена количество запросов будет не больше  $10^5$ , при этом количество запросов на добавление будет не больше 100.
- В тестах Марка количество запросов будет не больше  $10^5$ .

Постарайтесь, чтобы как можно больше программистов остались довольными. Билл оценивает корректность решения и понятность написанного вами кода. А Стивену и Марку кроме этого кажутся очень важными скорость работы решения и эффективность использования памяти (в том числе по сравнению с решениями других кандидатов). Подробные правила, по которым они будут вместе выставлять оценку, программисты пока разглашать не собираются...

## Задача 2. Generics

Для решения второй задачи кроме смекалки и аккуратности вам необходимо будет освоить (или вспомнить) два понятия языка C#, знакомство с которыми можно начать с MSDN:

1. Generics: [http://msdn.microsoft.com/en-us/library/ms379564\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms379564(VS.80).aspx).
2. Reflection: [http://msdn.microsoft.com/en-us/library/ms173183\(v=VS.100\).aspx](http://msdn.microsoft.com/en-us/library/ms173183(v=VS.100).aspx).

Представьте, что ваш коллега-зазнайка написал вот такой метод на языке C#:

```
Type CloseGeneric(Type openGeneric, Type concreteType)
{
    //...
}
```

Утверждается, что метод всегда, когда это возможно, возвращает тип данных `c`, для которого выполняются следующие два условия:

```
Type c = CloseGeneric(openGeneric, concreteType);
1. c.IsAssignableFrom(concreteType) == true
2. c.GetGenericTypeDefinition() == openGeneric
```

Если такого типа данных не существует, метод возвращает `null`.

Например, такой вызов:

```
CloseGeneric(typeof(IEnumerable<>), typeof(List<int>))
должен вернуть typeof(IEnumerable<int>).
```

А такой вызов:

```
CloseGeneric(typeof(IDictionary<,>), typeof(string))
должен вернуть null.
```

Коллега еще и поспорил с вами на шоколадку «Аленка», что вы не сможете найти в этом методе

ошибку. Правда, код показывать он отказывается, зато готов при вас запускать его на любых предложенных вами тестах.

Проигрывать шоколадку, а уж тем более «Аленку», совершенно неприемлемо! Вам нужно придумать набор тестов, который наверняка завалит решение этого задания. Приведите подробное описание тестовых случаев и комментарии к ним. Общие рассуждения о подходе к тестированию приветствуются.