Вступительные задания для кандидатов на стажировку по направлению .NET Backend Development

Основные моменты

Качество выполнения

- Оценивается как корректность выполненного задания, так и качество кода (читаемость, расширяемость, следование основным принципам ООП).

Инструменты

- Для выполнения задания по программированию вам понадобится интегрированная среда разработки Microsoft Visual Studio Community. Для работы с базами данных - SQL Server Management Studio. Рекомендуем английские версии.

Структура

- Решение каждого задания (C# проект, SQL файл) должно лежать в соответствующем номеру задания каталоге (Solution1, Solution2, и т.д.).

Способ сдачи

- Каталоги с решениями следует запаковать в один архив (*Имя Фамилия.zip)* и отправить на электронную почту dotnet-internship@mercurydevelopment.com

Задание #1.

Необходимо реализовать консольное приложение, которое будет предоставлять пользователю информацию о банковский кредитных договорах.

В качестве входного параметра приложение получает путь до файла (.json) со списком кредитных договоров. Кредитный договор может быть трех видов (автокредит, ипотека и кредит на образование) и определяется следующими полями:

- ID - уникальный номер кредита;
- Amount - сумма кредита;
- CountOfMonth - срок кредита в месяцах;
- Percent - процентная ставка кредита годовых;
- Borrower - информация о заемщике:
    - Id - уникальный номер заемщика;
    - FirstName - имя;
    - LastName - фамилия;
    - DateOfBirth - дата рождения;
    - PassportNumber - номер паспорта;
- Bank - информация о банке:
    - Id - уникальный номер банка;
    - Name - наименование банка;
    - Address - адрес банка;
- CarModel - модель автомобиля;
- CarBrand - марка автомобиля;
- VIN - VIN код автомобиля;
- AddressOfObject - адрес объекта ипотеки;
- Square - площадь объекта ипотеки;
- UniversityName - название университета;
- UniversityAddress - адрес университета.

Пользователь должен иметь возможность:

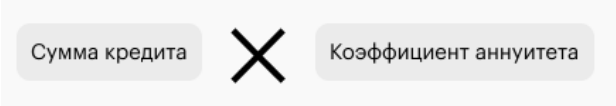
- Получить список всех кредитов вида:

```
ID кредита | Сумма кредита | Процентная ставка | Срок кредита | Тип кредита | Наименование банка | Фамилия и Имя заемщика
```

- Получить список всех банков;
- Получить список всех заемщиков;
- Получить список кредитов по заданному типу (автокредит, ипотека или кредит на образование);
- Добавить новый кредит;
- Получить список кредитов по заданной фамилии заемщика.
- Рассчитать сумму ежемесячного аннуитетного платежа для заданного кредита (по ID кредита).

NOTE: предполагается что вся информация будет хранится в оперативной памяти, без использования СУБД и прочих систем хранения. JSON файл с примерами кредитных договоров приложен к заданию.

NOTE 2: Формула расчета аннуитетного платежа:



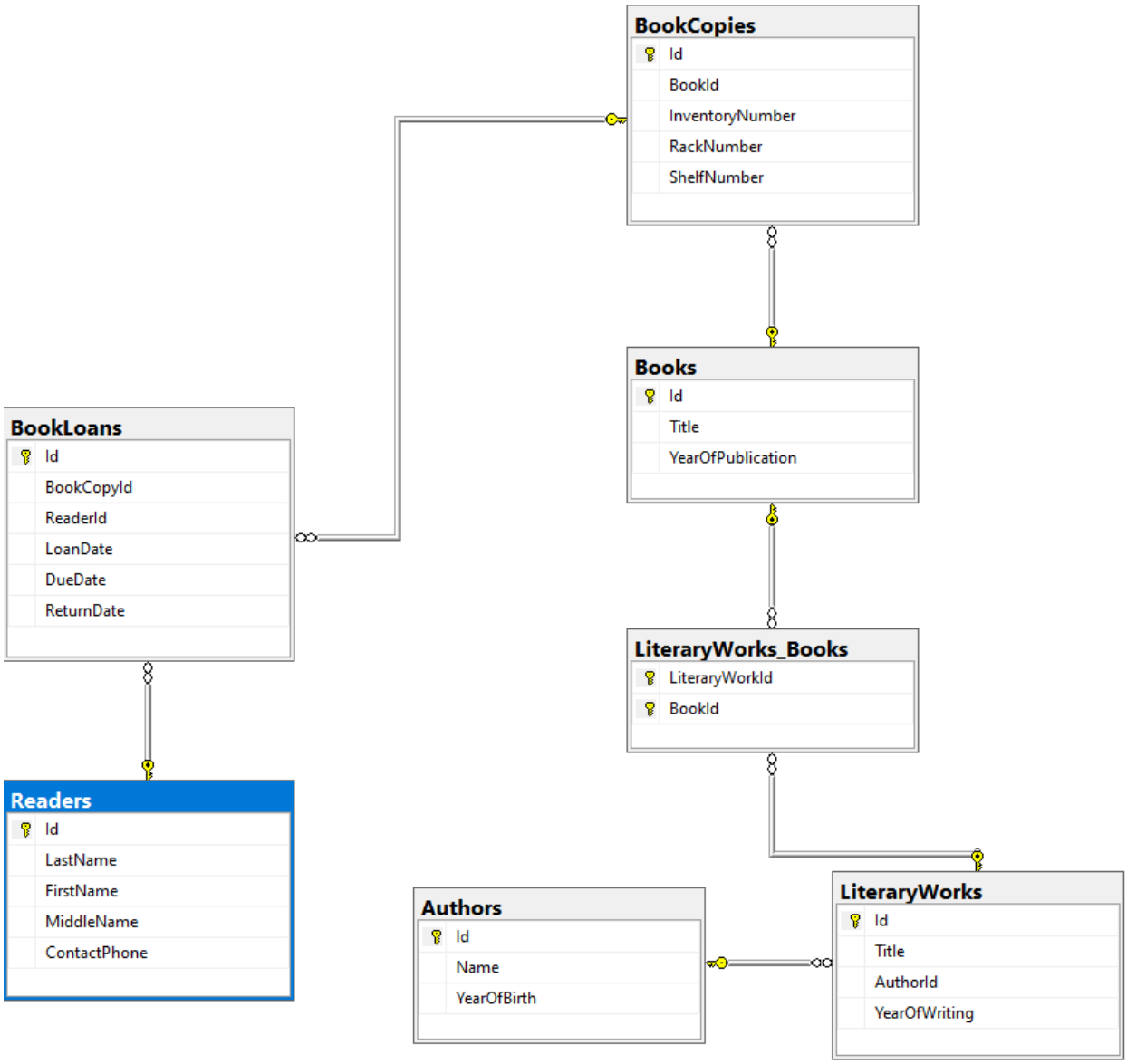
Формула расчета коэффициента аннуитета:

Для примера возьмем 300 000 рублей, срок 18 месяцев и процентную ставку 15% годовых.
Месячная процентная ставка = 15% / 12 = 1,25%, то есть 0,0125.
Количество платежей равно количеству месяцев — 18.
Коэффициент аннуитета:
0,0125 × (1 + 0,0125)18 / ((1 + 0,0125)18 − 1) = 0,062385.
Расчет платежа:
300 000 × 0,062385 = 18 715,44 ₽.

Задание #2. SQL

Имеется база данных “Библиотека”. Краткое описание:

- Книжные издания **Books** могут содержать одно (“отдельное издание”) или несколько литературных произведений (“сборники”). Литературные произведения хранятся в **LiteraryWorks**.
- Книжные издания могут иметь несколько бумажных экземпляров **BookCopies**, которые непосредственно содержатся на стеллажах с полками и выдаются читателям **Readers.**
- История выдачи отдельных экземпляров описана в таблице **BookLoans**, с датой выдачи **LoanDate** и датой фактического возврата **ReturnDate.**



Запрос 1.

Вывести список авторов и число написанных произведений. Отсортировать по убыванию числа. Результирующие колонки:

1. AuthorName
2. Count

Запрос 2.

Получить список читателей с количеством прочитанных литературных произведений. Считаем, что читатели читают все произведения в книгах, которые они брали. Перечитывание одного и того же произведения не считается. Сортировка от самого читающего к самому нечитающему. Результирующие колонки:

1. LastName
2. FirstName
3. Count

Запрос 3.

Вывести список литературных произведений, которые встречаются и в сборниках и в отдельных изданиях. Отсортировать по имени автора и году написания

1. Title
2. AuthorName
3. YearOfWriting

Запрос 4.

Вывести отчет по популярности авторов у читателей: автор, общее количество дней, когда его произведения были у читателей “на руках”, а также 3 самых популярных произведения

- Популярность надо рассчитать по общему количеству дней, когда литературные произведения были у читателей “на руках”.
- Для упрощения учитываются все экземпляры у всех читателей. Пересечения во времени не надо рассчитывать, т.е. даже если у одного и того же читателя одновременно находятся несколько экземпляров с одними и теми же произведениями - просто суммировать их
- В сборниках время равномерно распределяется среди всех произведений. Например, если в сборнике 4 произведения, и сборник был у читателя 8 дней, то каждое произведение “получает” 2 дня.

- Количество дней отсчитывается со следующего, т.е. если читатель вернул книгу в день получения - ничего не засчитываем
- 3 самых популярных произведения склеиваются и выводятся в отдельной колонке одной строкой в кавычках и разделенные запятыми. Пример:
  "Harry Potter and the Order of the Phoenix", "Harry Potter and the Half-Blood Prince", "Harry Potter and the Deathly Hallows"

Пример вывода пары строк (данные являются случайными, не стоит на них ориентироваться)

	AuthorName	CountAll	Top3
1	J.K. Rowling	101	"Harry Potter and the Order of the Phoenix", "Harry Potter and the Half-Blood Prince", "Harry Potter and the Deathly Hallows"
2	J.R.R. Tolkien	99	"The Hobbit", "The Two Towers", "The Silmarillion"

Задание #3. Алгоритм

Имеется текстовый документ **input.txt**, содержащий последовательность целых чисел от 0 до 99, которые разделены пробелом. Эти числа могут быть представлены в виде таблицы размером **N x N**, где **N** - число от 1 до 20.

1. В таком представлении чисел необходимо найти наибольшую сумму **M** подряд идущих чисел, расположенных в любом направлении (вверх, вниз, вправо, влево или по диагонали);
2. В таком представлении чисел необходимо найти наибольшее произведение **M** подряд идущих чисел, расположенных в любом направлении (вверх, вниз, вправо, влево или по диагонали).

Число **M** вводится пользователем и должно удовлетворять следующим условиям: $1 \leq M \leq 6$ и $M \leq N$.

Представленная ниже таблица наглядно показывает возможные направления для подсчёта при $N = 20$ и $M = 3$:

35 86 46 88 60 83 68 25 85 68 72 29 21 95 60 12 04 88 30 89
02 88 52 93 37 39 97 82 91 86 30 14 06 71 57 23 81 08 02 40
97 80 28 85 63 98 64 89 44 09 **60** 79 52 41 56 79 98 15 23 40
43 27 08 51 36 04 84 75 09 39 22 **88** 29 49 05 97 28 90 35 83
12 86 14 96 62 36 68 96 84 36 37 74 **15** 60 92 29 39 84 37 55
14 12 20 66 40 14 95 07 01 32 02 41 97 67 42 21 95 94 72 47
93 54 06 02 58 50 28 97 06 13 53 76 61 89 76 02 17 41 51 15
38 98 78 47 70 16 51 91 12 27 03 57 26 38 61 84 57 68 19 06
89 70 95 17 83 33 72 64 39 64 26 78 22 93 54 22 90 64 59 28
41 61 14 40 09 04 76 **77 04 39** 00 65 18 07 23 17 62 34 47 76
13 18 95 84 23 24 03 95 92 59 56 26 63 44 24 98 27 29 36 01
69 50 67 56 50 69 38 89 18 86 79 55 59 12 08 06 93 95 77 88
76 59 01 30 46 70 06 50 72 95 47 68 57 32 98 46 87 49 37 66
86 13 19 07 25 66 72 02 60 70 85 45 57 35 87 35 83 15 72 77
67 26 12 70 50 00 02 28 17 62 22 67 16 46 09 80 55 64 34 00
29 96 51 22 49 11 51 75 40 20 31 61 43 37 81 16 07 09 75 07
73 17 14 61 46 16 87 19 01 16 37 82 22 55 59 43 33 **48** 21 16
09 56 **92** 35 24 76 82 27 94 40 09 56 60 36 48 61 **30** 52 32 27
27 06 **43** 47 81 31 34 90 83 29 61 71 60 96 75 **92** 74 35 01 28
88 97 **81** 22 72 76 78 58 93 89 63 13 49 30 07 95 65 45 81 55

Задание #4. Интеграция с внешним API

Реализовать консольное приложение - **Обмен валюты.** Приложение интегрируется с внешним API (https://freecurrencyapi.com/). Необходимо зарегистрироваться и получить **API KEY**. (Используйте какую-нибудь стороннюю почту). **Запрещено использовать уже готовые библиотеки для решения задачи.**

Требования:

1. Создать интерфейс ***IExchangeService***, который содержит следующие методы:
    a. ***GetCurrencies()*** - получение всех валют.
    b. ***Exchange(from, to, amount)*** - перевод из одной валюты в другую.
    c. ***HistoricalExchange(from, to, amount, date)*** - перевод из одной валюты в другую на указанную дату в прошлом.
2. Создать класс ***ExchangeService***, который реализует ***IExchangeService***
3. **API KEY** должен передаваться в **Headers** запроса.
4. В качестве входных параметров пользователь вводит в консоль:
    a. 1 - Вывести все доступные валюты;
    b. 2 - Сделать обмен одной валюты в другую;
        i. Ввести код валюты из которой осуществляется обмен;
        ii. Ввести код валюты в которую осуществляется обмен;
        iii. Ввести число для обмена;
    c. 3 - Сделать обмен одной валюты в другую на указанную дату в прошлом
        i. Ввести код валюты из которой осуществляется обмен;
        ii. Ввести код валюты в которую осуществляется обмен;
        iii. Ввести число для обмена;
        iv. Ввести дату в формате **ГГГГ-ММ-ДД**
5. Все выходные данные следует также выводить на консоль;
6. Предусмотреть проверку на корректность пользовательского ввода;