Readme for Brute Force + Hashes

Этот проект реализует два алгоритма для поиска подстроки в строке:

1. **Метод грубой силы (Brute Force)**
2. **Метод с использованием хэширования**

**Для запуска программы не требуется никаких внешних подготовлений, строка и подстрока заданы внутри кода.  
  
Первая функция реализует Brute Force, вторая подход через хэши соответственно.**

**Тест 1:**

let S1 = "abcabdecab"; - исходная строка

let T1 = "cab"; - запрашиваемая подстрока

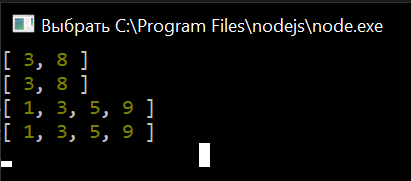
ожидаемый результат – 3, 8

Тест 2:

let S2 = "abababacaba";

let T2 = "aba";

Ожидаемый результат – 1, 3, 5, 9



1 и 3 строка – результат brute force, 2 и 4 – хэши.

**Основные функции**:

* findOccurrencesBruteForce(S, T) — реализует метод поиска подстроки с использованием brute-force.
  + Внутри происходит итерация по всем возможным позициям начала подстроки в строке.
  + Для каждой позиции проверяется, совпадают ли символы фрагмента строки с подстрокой T.
* findOccurrencesWithHash(S, T) — реализует метод поиска с использованием хэширования.
  + Для подстроки T вычисляется сумма кодов символов (хэш).
  + Затем для каждого фрагмента строки S с длиной, равной длине подстроки, вычисляется хэш, который сравнивается с хэшем подстроки.
  + Если хэши совпадают, то проводится дополнительная посимвольная проверка для исключения коллизий (когда два фрагмента могут иметь одинаковый хэш, но разные символы).

**Вычисление хэша**:

* Хэш вычисляется как сумма ASCII-кодов символов строки.
* Для первого фрагмента строки хэш вычисляется с нуля, а для всех последующих фрагментов используется рекуррентное вычисление: предыдущий хэш обновляется за счет удаления первого символа и добавления нового.

**Рекуррентное вычисление хэша**:

* Рекуррентное вычисление хэша ускоряет поиск, так как для каждого нового фрагмента строки хэш обновляется за константное время, вместо того чтобы вычислять его заново.

**Алгоритм**:

* В обоих методах поиск выполняется по строке S, начиная с каждого возможного индекса, где может начинаться подстрока T.
* В методе с хэшированием сначала сравниваются хэши, а затем, если хэши совпали, проводится посимвольная проверка на совпадение.
* В случае совпадения результат добавляется в массив positions.

**Задержка на выход**:

* В коде добавлена функция readline для создания задержки перед выходом программы. Это нужно для того, чтобы пользователь мог увидеть результат работы программы, прежде чем она завершится.

