Требуется написать прототип системы, которая следит за работой компьютерного клуба, обрабатывает события и подсчитывает выручку за день и время занятости каждого стола. Решение может быть реализовано на одном из языков C++ / Python / Golang.

Требования к решению

C++:
Решением задания будет: файл или несколько файлов с исходным кодом программы на языке C++ (стандарт до C++20 включительно), инструкции по сборке и тестовые примеры (количество тестов – на усмотрение разработчика).

Входные данные представляют собой текстовый файл. Файл указывается первым аргументом при запуске программы. Пример запуска программы:

```
$ task.exe test_file.txt
```

Программа должна компилироваться компилятором gcc или clang в Linux, mingw/cygwin в Windows. Рекомендуется использование стандартной библиотеки (STL). Использование любых сторонних библиотек, кроме STL, запрещено. В решении, кроме файлов с исходным кодом, требуется предоставить инструкции по компиляции программы для проверки. Можно, но необязательно, использовать следующие системы автоматизации сборки: make, automake, cmake, gradle; проприетарные средства сборки не допускаются: например, файлы проектов MS Visual Studio не подойдут.

Можно (но необязательно) написать проверочные юнит-тесты с использованием доступной библиотеки (Google Test, CppUTest, и т.п.).

Python:
Решением задания будет: файл или несколько файлов с исходным кодом программы на языке Python (версия 3.10 и старше), инструкции по запуску и тестовые примеры (количество тестов – на усмотрение разработчика).

Входные данные представляют собой текстовый файл. Файл указывается первым аргументом при запуске программы. Пример запуска программы:

```
$ python3 task.py test_file.txt
```

Программа должна запускать интерпретатором Python в Linux или Windows. Требуется использование стандартной библиотеки (Python Standard Library - PSL). Использование любых сторонних библиотек, кроме PSL, запрещено. В решении, кроме файлов с исходным кодом, требуется предоставить инструкции по запуску программы для проверки.

Можно (но необязательно) написать проверочные юнит-тесты с использованием доступных фреймворков (unittest, pytest, и т.п.).

Golang:
Решением задания будет: файл или несколько файлов с исходным кодом программы на языке Golang (версия 1.19 и старше) с использованием go modules, инструкции по запуску и тестовые примеры (количество тестов – на усмотрение разработчика).

Входные данные представляют собой текстовый файл. Файл указывается первым аргументом при запуске программы. Пример запуска программы:

```
$ task.exe test_file.txt
```

Программа должна запускаться в Linux или Windows с использованием docker container-а (требуется написание Dockerfile). Требуется использование стандартной библиотеки (https://pkg.go.dev/std). Использование любых сторонних библиотек, кроме стандартной, запрещено. В решении, кроме файлов с исходным кодом, требуется предоставить инструкции по запуску программы для проверки.

Формат входных данных
<количество столов в компьютерном клубе>
<время начала работы> <время окончания работы>
<стоимость часа в компьютерном клубе>
<время события 1> <идентификатор события 1> <тело события 1>
<время события 2> <идентификатор события 2> <тело события 2>
...
<время события N> <идентификатор события N> <тело события N>

Первая строка содержит количество столов в виде целого положительного числа.
Во второй строке задается время начала и окончания работы компьютерного клуба, разделенные пробелом.
В третьей строке задается стоимость часа в компьютерном клубе в виде целого положительного числа.
Затем задается список входящих событий, разделенных переносом строки. Внутри строки в качестве разделителя между элементами используется один пробел.

- Имена клиентов представляют собой комбинацию символов из алфавита a..z, 0..9, _, -
- Время задается в 24-часовом формате с двоеточием в качестве разделителя XX:XX, незначащие нули обязательны при вводе и выводе (например 15:03 или 08:09).
- Каждый стол имеет свой номер от 1 до N, где N – общее число столов, указанное в конфигурации.
- Все события идут последовательно во времени. (время события N+1) ≥ (время события N).

Выходные данные
Если входные данные не удовлетворяют описанному формату, программа должна вывести в консоль первую строку, в которой найдена ошибка формата и завершиться.

Если входные данные корректны, программа должна вывести следующий результат:

- На первой строке выводится время начала работы.
- Далее перечислены все события, произошедшие за рабочий день (входящие и исходящие), каждое на отдельной строке.
- После списка событий на отдельной строке выводится время окончания работы.
- Для каждого стола на отдельной строке выведены через пробел следующие параметры: Номер стола, Выручка за день и Время, которое он был занят в течение рабочего дня.

После этого программа должна завершиться.

Подсчет выручки

За каждый час, проведённый за столом, клиент платит цену, указанную в конфигурации. При оплате время округляется до часа в большую сторону, поэтому, даже если клиент занимал стол всего несколько минут, он платит за целый час. Выручка – сумма, полученная ото всех клиентов за всё время работы компьютерного клуба.

События

Все события характеризуются временем и идентификатором события. Исходящие события — это события, создаваемые во время работы программы. События, относящиеся к категории «входящие», сгенерированы быть не могут, и выводятся в том же виде, в котором были поданы во входном файле.

Входящие события:

ID 1. Клиент пришел

Формат: <время> 1 <имя клиента>

Если клиент уже в компьютерном клубе, генерируется ошибка "YouShallNotPass"

Если клиент пришел в нерабочие часы, тогда "NotOpenYet"

ID 2. Клиент сел за стол

Формат: <время> 2 <имя клиента> <номер стола>

Если клиент уже сидит за столом, то он может сменить стол.

Если стол <номер стола> занят (в том числе, если клиент пытается пересесть за стол, за которым сам и сидит), генерируется ошибка "PlaceIsBusy".

Если клиент не находится в компьютерном клубе, генерируется ошибка "ClientUnknown".

ID 3. Клиент ожидает

Формат: <время> 3 <имя клиента>

Если в клубе есть свободные столы, то генерируется ошибка "ICanWaitNoLonger!".

Если в очереди ожидания клиентов больше, чем общее число столов, то клиент уходит и генерируется событие ID 11.

ID 4. Клиент ушел

Формат: <время> 4 <имя клиента>

Если клиент не находится в компьютерном клубе, генерируется ошибка "ClientUnknown".

Когда клиент уходит, стол, за которым он сидел освобождается и его занимает первый клиент из очереди ожидания (ID 12).

Исходящие события:

ID 11. Клиент ушел

Формат: <время> 11 <имя клиента>
Генерируется в конце рабочего дня для всех клиентов, оставшихся в компьютерном клубе, **в алфавитном порядке их имен.** А также, когда клиент встает в очередь, а очередь ожидания уже заполнена.

ID 12. Клиент сел за стол

Формат: <время> 12 <имя клиента> <номер стола>
Генерируется для первого клиента в очереди при освобождении любого стола.

ID 13. Ошибка

Формат: <время> 13 <ошибка>
Выводится сразу после события, которое вызвало ошибку. Событие, вызвавшее ошибку, считается не выполненным, и никакого эффекта на состояние клиентов не оказывает.

Пример работы программы:

Входной файл	Вывод в консоль
3 09:00 19:00 10 08:48 1 client1 09:41 1 client1 09:48 1 client2 09:52 3 client1 09:54 2 client1 1 10:25 2 client2 2 10:58 1 client3 10:59 2 client3 3 11:30 1 client4 11:35 2 client4 2 11:45 3 client4 12:33 4 client1 12:43 4 client2 15:52 4 client4	09:00 08:48 1 client1 08:48 13 NotOpenYet 09:41 1 client1 09:48 1 client2 09:52 3 client1 09:52 13 ICanWaitNoLonger! 09:54 2 client1 1 10:25 2 client2 2 10:58 1 client3 10:59 2 client3 3 11:30 1 client4 11:35 2 client4 2 11:35 13 PlaceIsBusy 11:45 3 client4 12:33 4 client1 12:33 12 client4 1 12:43 4 client2 15:52 4 client4 19:00 11 client3 19:00 1 70 05:58 2 30 02:18 3 90 08:01