Задача

Написать программу mtfind, производящую поиск подстроки в текстовом файле по маске с использованием многопоточности.

Маска - это строка, где "?" обозначает любой символ.

Программа принимает в качестве параметров командной строки:

1. Имя текстового файла, в котором должен идти поиск (размер файла - до 1Гб).
2. Маску для поиска, в кавычках. Максимальная длина маски 1000 символов.

Вывод программы должен быть в следующем формате:

- На первой строке - количество найденных вхождений.
- Далее информация о каждом вхождении, каждое на отдельной строке, через пробел: номер строки, позиция в строке, само найденное вхождение.
- Порядок вывода найденных вхождений должен совпадать с их порядком в файле
- Вся нумерация ведется начиная с 1 (делаем программу для обычных людей)

Дополнения:

- В текстовом файле кодировка только 7-bit ASCII
- Поиск с учетом регистра
- Каждое вхождение может быть только на одной строке. Маска не может содержать символа перевода строки
- Найденные вхождения не должны пересекаться. Если в файле есть пересекающиеся вхождения то нужно вывести одно из них (любое).
- Пробелы и разделители участвуют в поиске наравне с другими символами.
- Можно использовать STL, Boost, возможности C++1x, C++2x.
- Многопоточность нужно использовать обязательно. Однопоточные решения засчитываться не будут.
- Серьезным плюсом будет разделение работы между потоками равномерно вне зависимости от количества строк во входном файле.

Решение представить в виде архива с исходным кодом и проектом CMake или Visual Studio (либо в виде ссылки на онлайн Git-репозиторий). Код должен компилироваться в GCC или MSVC.

ПРИМЕР

Файл input.txt:

```
I've paid my dues
Time after time.
I've done my sentence
But committed no crime.
And bad mistakes ?
I've made a few.
I've had my share of sand kicked in my face
But I've come through.
```

Запуск программы: `mtfind input.txt "?ad"`

Ожидаемый результат:

```
3
5 5 bad
6 6 mad
7 6 had
```

Критерии оценки решения:

- Правильность выдаваемых результатов

- Качество и читабельность кода, легкость дальнейшего развития и поддержки
- Скорость работы и потребление памяти

Все критерии одинаково важны