Без году неделя

Язык: F#
Максимальное количество баллов: 16
Напишите 14 функций (и тесты для них) для обработки календарных дат. Во всех заданиях под датой подразумевается значение типа **`(int * int * int)`**, где первый элемент - это год, второй - месяц, третий - день. Для доступа к отдельным элементам кортежа используйте следующие функции:

```
let year (a, _, _) = a
let month (_, a, _) = a
let day (_, _, a) = a
```

Ваше решение обязано работать только для корректных дат, удовлетворяющих следующим ограничениям:

```
year > 0
1 <= month <= 12
1 <= day <= 31 (правая граница зависит от месяца)
1 <= day_of_year <= 365
```

Проверять даты на корректность и обрабатывать високосные года (кроме последней задачи) не требуется.

1. (+1) Напишите функцию **`is_older`**, которая принимает две даты и возвращает **`true`** либо **`false`**. **`true`** нужно возвращать только в том случае, если первый аргумент представляет собой более раннюю дату, чем второй.
2. (+1) Напишите функцию **`number_in_month`**, которая принимает список дат и номер месяца (целое число) и возвращает количество дат из списка, приходящихся на этот месяц.
3. (+1) Напишите функцию **`number_in_months`**, которая принимает список дат и список номеров месяцев (список целых чисел), и возвращает количество дат из списка, которые приходятся на любой месяц из списка. Считайте, что номера месяцев не повторяются.
4. (+1) Напишите функцию **`dates_in_month`**, которая принимает список дат и номер месяца (целое число), и возвращает список,

который содержит все даты из исходного списка, которые приходятся на этот месяц. Порядок дат сохраните.

5. (+1) Напишите функцию **`dates_in_months`**, которая принимает список дат и список номеров месяцев (список целых чисел) и возвращает список дат, которые приходятся на любой из исходных месяцев. Считайте, что номера месяцев не повторяются. Подсказка: воспользуйтесь оператором склеивания списков `@`
6. (+1) Напишите функцию **`get_nth`**, которая принимает список строк и целое число **`n`**, и возвращает n-й элемент списка (нумерация начинается с 1). Считайте, что количество элементов списка не меньше n.
7. (+1) Напишите функцию **`date_to_string`**, которая принимает дату и возвращает строку в виде **`"January 19, 1991"`**. Для конвертирования числа в строку используйте **`ToString`**, например **`x.ToString`**
8. (+1) Напишите функцию **`number_before_reaching_sum`**, которая принимает целое число **`sum`** (считайте, что оно положительно), и список целых чисел (считайте, что список содержит только положительные числа), и возвращает число n, такое, что сумма первых **`n`** элементов списка меньше **`sum`**, а сумма первых **`n + 1`** элементов списка - больше или равна **`sum`**. Считайте, что сумма всех элементов списка больше **`sum`**.
9. (+1) Напишите функцию **`what_month`**, которая принимает день в году (целое число от 1 до 365), и возвращает номер месяца, на который приходится этот день (1 для января, 2 для февраля и т. д.)
10. (+1) Напишите функцию **`month_range`**, которая принимает два дня в году **`day1`** и **`day2`** и возвращает список целых чисел **`[m1; m2 … mn]`**, где **`m1`** - месяц, на который приходится **`day1`**, **`m2`** - месяц, на который приходится **`day1 + 1, …`** **`mn`** - месяц, на который приходится **`day2`**. Длина результирующего списка должна быть равно **`day2 - day1 + 1`**, если **`day2 >= day1`**, и 0 в противном случае.
11. (+1) Напишите функцию **`oldest`**, которая принимает список дат и возвращает **`(int * int * int) option`**: **`None`**, если исходный список пуст, или **`Some d`**, где **`d`** - наиболее ранняя дата в списке.

12. (+2) Напишите функции **`number_in_months_2`** и **`dates_in_months_2`**, аналогичные функциям из задач 3 и 5, но корректно обрабатывающие случай, когда месяцы в списке могут повторяться. Подсказка: напишите функцию, удаляющую дубликаты, а потом используйте предыдущее решение.
13. (+3) Напишите функцию **`reasonable_date`**, которая принимает на вход дату и возвращает **`true`**, если аргумент является существующей датой из нашей эры: положительный год, месяц из диапазона **`[1, 12]`**, день в границах, допустимых для данного месяца. Не забудьте про високосный год (переход на Григорианский календарь обрабатывать не требуется)

В результате, в файле с решением должны быть следующие функции:

```
val is_older : (int * int * int) * (int * int * int) -> bool
val number_in_month : (int * int * int) list * int -> int
val number_in_months : (int * int * int) list * int list ->
int
val dates_in_month : (int * int * int) list * int -> (int *
int * int) list
val dates_in_months : (int * int * int) list * int list ->
(int * int * int) list
val get_nth : string list * int -> string
val date_to_string : int * int * int -> string
val number_before_reaching_sum : int * int list -> int
val what_month : int -> int
val month_range : int * int -> int list
val oldest : (int * int * int) list -> (int * int * int)
option
val number_in_months_2 : (int * int * int) list * int list ->
int
val dates_in_months_2 : (int * int * int) list * int list ->
(int * int * int) list
val reasonable_date : int * int * int -> bool
```