Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 2 з дисципліни
«Мультипарадигменне програмування»

«Функціональне програмування»

Виконав: Германович Д.А. ІП-02

Київ 2022

Завдання

Ви напишете 11 функцій SML (і тести для них), пов'язаних з календарними датами. У всіх завданнях, **"дата"** є значенням SML типу int*int*int, де перша частина - це рік, друга частина - місяць і третя частина - день. **«Правильна»** дата має позитивний рік, місяць від 1 до 12 і день не більше 31 (або 28, 30 - залежно від місяця). Перевіряти "правильність" дати не обов'язково, адже це досить складна задача, тож будьте готові до того, що багато ваших функцій будуть працювати корректно для деяких/всіх **"неправильних"** дат у тому числі. Також, **«День року»** — це число від 1 до 365 де, наприклад, 33 означає 2 лютого. (Ми ігноруємо високосні роки, за винятком однієї задачі.)

1. Напишіть функцію is_older, яка приймає дві дати та повертає значення true або false. Оцінюється як true, якщо перший аргумент - це дата, яка раніша за другий аргумент. (Якщо дві дати однакові, результат хибний.)
2. Напишіть функцію number_in_month, яка приймає список дат і місяць (тобто int) і повертає скільки дат у списку в даному місяці.
3. Напишіть функцію number_in_months, яка приймає список дат і список місяців (тобто список int) і повертає кількість дат у списку дат, які знаходяться в будь-якому з місяців у списку місяців. Припустимо, що в списку місяців немає повторюваних номерів. Підказка: скористайтеся відповіддю до попередньої задачі.
4. Напишіть функцію dates_in_month, яка приймає список дат і число місяця (тобто int) і повертає список, що містить дати з аргументу "список дат", які знаходяться в переданому місяці. Повернутий список повинен містять дати в тому порядку, в якому вони були надані спочатку.
5. Напишіть функцію dates_in_months, яка приймає список дат і список місяців (тобто список int) і повертає список, що містить дати зі списку аргументів дат, які знаходяться в будь-якому з місяців у списку місяців. Для простоти, припустимо, що в списку місяців немає повторюваних номерів. Підказка: Використовуйте свою відповідь на попередню задачу та оператор додавання списку SML (@).
6. Напишіть функцію get_nth, яка приймає список рядків і int n та повертає n-й елемент списку, де голова списку є першим значенням. Не турбуйтеся якщо в списку занадто мало елементів: у цьому випадку ваша функція може навіть застосувати hd або tl до порожнього списку, і це нормально.
7. Напишіть функцію date_to_string, яка приймає дату і повертає рядок у вигляді "February 28, 2022" Використовуйте оператор ^ для конкатенації рядків і бібліотечну функцію Int.toString для перетворення int в рядок. Для створення частини з місяцем не використовуйте купу розгалужень. Замість цього використайте список із 12 рядків і свою відповідь на попередню задачу. Для консистенції пишіть кому після дня та використовуйте назви місяців англійською мовою з великої літери.

8. Напишіть функцію number_before_reaching_sum, яка приймає додатний int під назвою sum, та список int, усі числа якої також додатні. Функція повертає int. Ви повинні повернути значення int n таке, щоб перші n елементів списку в сумі будуть менші sum, але сума значень від n + 1 елемента списку до кінця був більше або рівний sum.
9. Напишіть функцію what_month, яка приймає день року (тобто int між 1 і 365) і повертає в якому місяці цей день (1 для січня, 2 для лютого тощо). Використовуйте список, що містить 12 цілих чисел і вашу відповідь на попередню задачу.
10. Напишіть функцію month_range, яка приймає два дні року day1 і day2 і повертає список int [m1,m2,...,mn] де m1 – місяць day1, m2 – місяць day1+1, ..., а mn – місяць day2. Зверніть увагу, що результат матиме довжину day2 - day1 + 1 або довжину 0, якщо day1>day2.

11. Напишіть найстарішу функцію, яка бере список дат і оцінює параметр (int*int*int). Він має оцінюватися як NONE, якщо список не містить дат, і SOME d, якщо дата d є найстарішою датою у списку.

Реалізація

```
(*task 1*)
fun is_older (date1: int*int*int, date2: int*int*int) =
 if (#1 date1) > (#1 date2)
 then false
 else if (#2 date1) > (#2 date2)
 then false
 else if (#3 date1) > (#3 date2)
 then false
 else true

fun provided_test1 () =
    let val date1 = (20,6,2003)
        val date2 = (21,6,2005)
    in
        is_older(date1,date2)
    end


val test1 = provided_test1();

(*task 2*)
fun number_in_month (dates: (int*int*int) list, month: int) =
     List.length (List.filter (fn date => #2 date = month) dates);

fun provided_test2 () =
    let val dates =
[(2004,2,9),(2003,3,23),(2006,4,30),(2006,12,15),(2006,3,1)];
        val month = 3;
    in
        number_in_month(dates,month)
    end

val test2 = provided_test2();

(*task 3*)

fun number_in_months
     (dates: (int*int*int) list, months: int list) =
          if List.length months = 0 then 0
          else number_in_months(dates, tl months) + number_in_month(dates, hd
months);

fun provided_test3 () =
    let val dates =
[(2004,2,9),(2003,3,23),(2006,4,30),(2014,4,12),(2006,12,15),(2006,3,1)];
        val months = [3,4,5];
    in
        number_in_months(dates,months)
    end

val test3 = provided_test3();
```

```
(*task 4*)

fun dates_in_month (dates: (int*int*int) list, month: int) =
     List.filter (fn date => #2 date = month) dates;

fun provided_test4 () =
    let val dates =
[(2004,2,9),(2003,3,23),(2006,4,30),(2006,12,15),(2006,3,1)];
         val month = 3;
    in
         dates_in_month(dates,month)
    end

val test4 = provided_test4();

(*task 5*)


fun dates_in_months
     (dates: (int*int*int) list, months: int list) =
          if List.length months = 0 then []
          else dates_in_months(dates, tl months) @ dates_in_month(dates, hd
months);

fun provided_test5 () =
    let val dates =
[(2004,2,9),(2003,3,23),(2006,4,30),(2014,4,12),(2006,12,15),(2006,3,1)];
         val months = [3,4,5];
    in
         dates_in_months(dates,months)
    end

val test5 = provided_test5();

(*task 6*)

fun get_nth (strings: string list, n: int) =
     if n = 1 then hd strings
     else get_nth(tl strings, n - 1 );

fun provided_test6 () =
    let val strings =
["(1)What","(2)is","(3)the","(4)meaning","(5)of","(6)life,","(6)the","(7)univers
e",
    "(8)and","(9)everything?"];
         val n = 5;
    in
         get_nth(strings, n)
    end

val test6 = provided_test6();
```

```
(*task 7*)
val months = ["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"];

fun date_to_string(date: int*int*int) = get_nth(months,#2 date) ^ " " ^
     Int.toString (#3 date) ^ ", " ^ Int.toString (#1 date);

fun provided_test7 () =
     let
          val date = (2022, 2, 28);
     in
          date_to_string(date)
     end

val test7 = provided_test7();

(*task 8*)

fun number_before_reaching_sum (sum: int, number_list:int list) =
     if List.length (number_list) = 0 then 0
     else if hd(number_list) < sum
     then 1 + number_before_reaching_sum(sum,((hd(number_list)
          + hd(tl(number_list))))::(tl(tl(number_list)))))
     else 0;

fun provided_test8 () =
     let
          val sum = 6;
          val number_list = [1, 1, 3, 2, 1, 1, 1, 1];
     in
          number_before_reaching_sum(sum,number_list)
     end;

val test8 = provided_test8();

(*task 9*)

val months_to_days = [  31,
28,  31,  30,  31,  30,  31,  31,  30,  31,  30,  31];

fun what_month (day: int) = number_before_reaching_sum(day, months_to_days) + 1;

fun provided_test9 () =
     let
          val day = 68;
     in
          what_month(day)
     end;

val test9 = provided_test9();
```

```
(*task 10*)

fun month_range(day1: int, day2: int) =
if day1 > day2 then []
else what_month(day1) :: month_range(day1 +1, day2);

fun provided_test10 () =
     let
          val day1 = 25;
          val day2 = 35;
     in
          month_range(day1, day2)
     end;

val test10 = provided_test10();

(*task 11*)

fun get_oldest_date (dates: (int*int*int) list) =
     if List.length dates = 0 then NONE
     else if List.length dates = 1 then SOME(hd dates)
     else if is_older(hd dates, hd(tl dates)) then
get_oldest_date(hd(dates)::tl(tl(dates)))
     else get_oldest_date(hd(tl(dates))::tl(tl(dates)));

fun provided_test11 () =
     let
          val dates =
[(12,12,2013),(12,12,2012),(12,12,2010),(12,12,2020),(12,12,2015)];
     in
          get_oldest_date(dates)
     end;

val test11 = provided_test11();
```

Результат виконання

Task1

```
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val provided_test1 = fn : unit -> bool
val test1 = true : bool
```

Task2

```
val number_in_month = fn : (int * int * int) list * int -> int

val provided_test2 = fn : unit -> int
val test2 = 2 : int
```

Task3

```
val number_in_months = fn : (int * int * int) list * int list -> int

val provided_test3 = fn : unit -> int
val test3 = 4 : int
```

Task4

```
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list

val provided_test4 = fn : unit -> (int * int * int) list
val test4 = [(2003,3,23),(2006,3,1)] : (int * int * int) list
```

Task5

```
val dates_in_months = fn :
  (int * int * int) list * int list -> (int * int * int) list

val provided_test5 = fn : unit -> (int * int * int) list
val test5 = [(2006,4,30),(2014,4,12),(2003,3,23),(2006,3,1)] :
  (int * int * int) list
```

Task6

```
val get_nth = fn : string list * int -> string

val provided_test6 = fn : unit -> string
val test6 = "(5)of" : string
```

Task7

```
val months =
  ["January","February","March","April","May","June","July","August",
   "September","October","November","December"] : string list

val date_to_string = fn : int * int * int -> string

val provided_test7 = fn : unit -> string
val test7 = "February 28, 2022" : string
```

Task8

```
val number_before_reaching_sum = fn : int * int list -> int

val provided_test8 = fn : unit -> int

val test8 = 3 : int
```

Task9

```
val months_to_days = [31,28,31,30,31,30,31,31,30,31,30,31] : int list

val what_month = fn : int -> int

val provided_test9 = fn : unit -> int

val test9 = 3 : int
```

Task10

```
val month_range = fn : int * int -> int list

val provided_test10 = fn : unit -> int list

val test10 = [1,1,1,1,1,1,1,2,2,2,2] : int list
```

Task11

```
val provided_test11 = fn : unit -> (int * int * int) option

val test11 = SOME (12,12,2010) : (int * int * int) option
```

Висновок

В результаті виконання лабороторної работи я вивчив базові конструкції функціональної мови програмування SML, реалізував та протестував 11 функцій