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: ІП-02 Ілющенко В. Т.

Київ 2022

Лабораторна робота 2

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

Завдання

Для зручності подальшої роботи спочатку створимо тип date (int*int*int), який будемо використовувати у значній кількості функцій. Далі розробимо функції, що вимагаються у завданні, з урахуванням особливостей мови програмування SML:

Реалізація

```
(*This file contains the functions that were required by the laboratory work.
The tests for these functions can be found in the index.sml file *)

type date = int*int*int;

fun is_older (date_first : date, date_second : date) =
    if (#1 date_first) <> (#1 date_second)
    then (#1 date_first) < (#1 date_second)
    else

    if (#2 date_first) <> (#2 date_second)
    then (#2 date_first) < (#2 date_second)
    else

    (#3 date_first) < (#3 date_second);

fun number_in_month (dates : date list, month_num : int) =
    if null dates
    then 0
    else

    if (#2 (hd dates)) = month_num
    then 1 + number_in_month(tl dates, month_num)
    else number_in_month(tl dates, month_num);

fun number_in_months (dates : date list, month_nums : int list) =
    if null dates orelse null month_nums
    then 0
    else

    number_in_month(dates, hd month_nums) + number_in_months(dates, tl
month_nums);

fun dates_in_month (dates : date list, month_num : int) =
    if null dates
    then []
    else

    if (#2 (hd dates)) = month_num
    then (hd dates) :: dates_in_month(tl dates, month_num)
```

```
    else dates_in_month(tl dates, month_num);

fun dates_in_months (dates : date list, month_nums : int list) =
    if null dates orelse null month_nums
    then []
    else

    dates_in_month(dates, hd month_nums) @ dates_in_months(dates, tl month_nums);

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

fun date_to_string(input_date : date) =
    let val months = [ "January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"]
    in get_nth(months, #2 input_date) ^ " " ^ Int.toString(#3 input_date) ^ ", "
^ Int.toString(#1 input_date)
    end;

fun number_before_reaching_sum(sum : int, numbers: int list) =
    if null numbers
    then 0
    else

    if sum <= hd numbers
    then 0
    else 1 + number_before_reaching_sum((sum - (hd numbers)), (tl numbers));

fun what_month(day : int) =
    let val days_in_months = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    in number_before_reaching_sum(day, days_in_months) + 1
    end;

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


fun oldest(dates : date list) =
    if null dates
    then NONE
    else

    let val tl_oldest = oldest(tl dates)
    in if isSome tl_oldest andalso is_older(valOf tl_oldest, hd dates)
    then tl_oldest
    else SOME (hd dates)
    end;
```

```
use "functions.sml";

fun is_older_test() =
    if is_older((2022,5,4), (2022,5,4)) <> false
    then raise Fail "Test failed at 1 case (is_older)"
    else

    if is_older((2022,5,4), (2022,5,5)) <> true
    then raise Fail "Test failed at 2 case (is_older)"
    else

    if is_older((2022,6,4), (2022,5,5)) <> false
    then raise Fail "Test failed at 3 case (is_older)"
    else

    print("Test passed (is_older)\n");

fun number_in_month_test() =
    if number_in_month([(2022,2,28), (2022,2,28), (2022,6,28)], 2) <> 2
    then raise Fail "Test failed at 1 case (number_in_month)"
    else

    if number_in_month([(2022,2,28), (2022,2,28), (2022,2,28)], 5) <> 0
    then raise Fail "Test failed at 2 case (number_in_month)"
    else

    if number_in_month([(2022,3,28), (2022,2,28), (2022,3,28)], 3) <> 2
    then raise Fail "Test failed at 3 case (number_in_month)"
    else

    print("Test passed (number_in_month)\n");

fun number_in_months_test() =
    if number_in_months([(2022,2,28), (2022,3,28), (2022,6,28)], [2, 3]) <> 2
    then raise Fail "Test failed at 1 case (numbers_in_month)"
    else

    if number_in_months([(2022,2,28), (2022,3,28), (2022,6,28)], [7, 8, 9]) <> 0
    then raise Fail "Test failed at 2 case (numbers_in_month)"
    else

    if number_in_months([(2022,2,28), (2022,3,28), (2022,6,28)], [3, 6]) <> 2
    then raise Fail "Test failed at 3 case (numbers_in_month)"
    else

    print("Test passed (numbers_in_month)\n");

fun dates_in_month_test() =
```

```
    if dates_in_month([(2022,2,28), (2022,2,28), (2022,6,28)], 2) <>
[(2022,2,28), (2022,2,28)]
    then raise Fail "Test failed at 1 case (dates_in_month)"
    else

    if dates_in_month([(2022,2,28), (2022,2,28), (2022,6,28)], 7) <> []
    then raise Fail "Test failed at 2 case (dates_in_month)"
    else

    if dates_in_month([(2023,7,28), (2027,9,8), (2022,7,28)], 7) <> [(2023,7,28),
(2022,7,28)]
    then raise Fail "Test failed at 3 case (dates_in_month)"
    else

    print("Test passed (dates_in_month)\n");

fun dates_in_months_test() =
    if dates_in_months([(2022,2,28), (2022,2,28), (2022,6,28), (2022,7,28)], [2,
6]) <> [(2022,2,28), (2022,2,28), (2022,6,28)]
    then raise Fail "Test failed at 1 case (dates_in_month)"
    else

    if dates_in_months([(2022,2,28), (2022,2,28), (2022,6,28), (2022,7,28)], [])
<> []
    then raise Fail "Test failed at 2 case (dates_in_month)"
    else

    if dates_in_months([(2023,7,28), (2027,9,8), (2022,7,28)], [7]) <>
[(2023,7,28), (2022,7,28)]
    then raise Fail "Test failed at 3 case (dates_in_month)"
    else

    print("Test passed (dates_in_months)\n");

fun get_nth_test() =
    if get_nth(["A", "B", "C", "D"], 2) <> "B"
    then raise Fail "Test failed at 1 case (get_nth)"
    else

    if get_nth(["A", "B", "C", "D"], 4) <> "D"
    then raise Fail "Test failed at 2 case (get_nth)"
    else

    if get_nth(["A", "B", "C", "D"], 1) <> "A"
    then raise Fail "Test failed at 3 case (get_nth)"
    else

    print("Test passed (get_nth)\n");

fun date_to_string_test() =
    if date_to_string((2022,2,28)) <> "February 28, 2022"
```

```
    then raise Fail "Test failed at 1 case (date_to_string)"
    else

    if date_to_string((2012,12,21)) <> "December 21, 2012"
    then raise Fail "Test failed at 2 case (date_to_string)"
    else

    if date_to_string((2022,2,24)) <> "February 24, 2022"
    then raise Fail "Test failed at 3 case (date_to_string)"
    else

    print("Test passed (date_to_string)\n");

fun number_before_reaching_sum_test() =
    if number_before_reaching_sum(10, [1, 3, 9, 11, 22, 33]) <> 2
    then raise Fail "Test failed at 1 case (number_before_reaching_sum)"
    else

    if number_before_reaching_sum(1, [1, 3, 9, 11, 22, 33]) <> 0
    then raise Fail "Test failed at 2 case (number_before_reaching_sum)"
    else

    if number_before_reaching_sum(14, [1, 3, 9, 11, 22, 33]) <> 3
    then raise Fail "Test failed at 3 case (number_before_reaching_sum)"
    else

    print("Test passed (number_before_reaching_sum)\n");

fun what_month_test() =
    if what_month(32) <> 2
    then raise Fail "Test failed at 1 case (what_month)"
    else

    if what_month(1) <> 1
    then raise Fail "Test failed at 2 case (what_month)"
    else

    if what_month(70) <> 3
    then raise Fail "Test failed at 3 case (what_month)"
    else

    print("Test passed (what_month)\n");

fun month_range_test() =
    if month_range(30, 32) <> [1, 1, 2]
    then raise Fail "Test failed at 1 case (month_range)"
    else

    if month_range(32, 30) <> []
    then raise Fail "Test failed at 2 case (month_range)"
    else
```

```
    if month_range(1, 2) <> [1, 1]
    then raise Fail "Test failed at 3 case (month_range)"
    else

    print("Test passed (month_range)\n");

fun oldest_test() =
    if oldest([(2021,3,7), (2024,10,8), (2021,6,2), (2011,3,7)]) <> SOME
(2011,3,7)
    then raise Fail "Test failed at 1 case (oldest)"
    else

    if oldest([]) <> NONE
    then raise Fail "Test failed at 2 case (oldest)"
    else

    if oldest([(1024,1,8), (2074,12,8), (2224,4,2), (2020,1,1)]) <> SOME
(1024,1,8)
    then raise Fail "Test failed at 3 case (oldest)"
    else

    print("Test passed (oldest)\n");

val _ = is_older_test();
val _ = number_in_month_test();
val _ = number_in_months_test();
val _ = dates_in_month_test();
val _ = dates_in_months_test();
val _ = get_nth_test();
val _ = date_to_string_test();
val _ = number_before_reaching_sum_test();
val _ = what_month_test();
val _ = month_range_test();
val _ = oldest_test();
```

Результат виконання програми

```
D:\Temp\mp-lab-2>REM
Standard ML of New Jersey (32-bit) v110.99.2 [built: Tue Sep 28 13:04:14 2021]
[opening .\index.sml]
[opening functions.sml]
type date = int * int * int
val is_older = fn : date * date -> bool
val number_in_month = fn : date list * int -> int
val number_in_months = fn : date list * int list -> int
val dates_in_month = fn : date list * int -> date list
val dates_in_months = fn : date list * int list -> date list
val get_nth = fn : string list * int -> string
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
[autoloading done]
val date_to_string = fn : date -> string
val number_before_reaching_sum = fn : int * int list -> int
val what_month = fn : int -> int
val month_range = fn : int * int -> int list
val oldest = fn : date list -> date option
val it = () : unit
val is_older_test = fn : unit -> unit
val number_in_month_test = fn : unit -> unit
val number_in_months_test = fn : unit -> unit
val dates_in_month_test = fn : unit -> unit
val dates_in_months_test = fn : unit -> unit
val get_nth_test = fn : unit -> unit
val date_to_string_test = fn : unit -> unit
val number_before_reaching_sum_test = fn : unit -> unit
val what_month_test = fn : unit -> unit
val month_range_test = fn : unit -> unit
val oldest_test = fn : unit -> unit
Test passed (is_older)
Test passed (number_in_month)
Test passed (numbers_in_month)
Test passed (dates_in_month)
Test passed (dates_in_months)
Test passed (get_nth)
Test passed (date_to_string)
Test passed (number_before_reaching_sum)
Test passed (what_month)
Test passed (month_range)
Test passed (oldest)
-
```

Висновки

Під час виконання даної лабораторної роботи було реалізовано 11 функцій з використанням функціональної мови програмування SML. Ми ознайомились з функціональною парадигмою програмування та побачили її відмінності від імперативної, яку ми використовували під час минулої лабораторної роботи. Також нами було розроблено тести для реалізованих функцій, під час роботи яких ми переконалися у коректності роботи розроблених функцій.