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»

Виконав(ла) *ІП-01 Пашковський Євгеній Сергійович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *ас. Очеретяний О.К.*
(прізвище, ім'я, по батькові)

Київ 2022

1. Завдання лабораторної роботи

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, ..., a 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.

2. Опис використаних технологій

Під час виконання лабораторної роботи було задіяно мову SML (Standard ML) за умовою завдання. Було використано редактор коду Visual Studio Code.

3. Вихідний код

```
(* 1 *)
fun is_older (date1 : int*int*int, date2 : int*int*int) =
((#1 date1) < (#1 date2)) orelse
((#1 date1) = (#1 date2) andalso (#2 date1) < (#2 date2)) orelse
((#1 date1) = (#1 date2) andalso (#2 date1) = (#2 date2) andalso (#3 date1) < (#3 date2));
(* 2 *)
fun number_in_month ([], month : int, acc : int) = acc |
number_in_month ((x : int*int*int)::xs, month : int, acc : int) = number_in_month (xs,
month, acc + (if (#2 x) = month then 1 else 0));
(* 3 *)
fun number_in_months (list_of_dates : (int*int*int) list, [], acc) = acc |
number_in_months (list_of_dates : (int*int*int) list, (x : int)::xs, acc) =
number_in_months(list_of_dates, xs, acc + number_in_month(list_of_dates, x, 0));
(* 4 *)
fun dates_in_month ([], month : int) = [] |
dates_in_month ((x : int*int*int)::xs, month : int) = if (#2 x) = month then ([x] @
dates_in_month(xs, month)) else dates_in_month(xs, month);
(* 5 *)
fun dates_in_months (list_of_dates : (int*int*int) list, []) = [] |
dates_in_months (list_of_dates : (int*int*int) list, (x : int)::xs) =
dates_in_month(list_of_dates, x) @ dates_in_months(list_of_dates, xs);
(* 6 *)
fun get_nth (list_of_strings : string list, 1) = hd list_of_strings |
get_nth (list_of_strings : string list, n : int) = get_nth(tl list_of_strings, n - 1);
(* 7 *)
val months_names = ["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"];
fun date_to_string (date : int*int*int) = get_nth(months_names, #2 date) ^ " " ^
Int.toString (#3 date) ^ ", " ^ Int.toString (#1 date);
(* 8 *)
fun number_before_reaching_sum (sum : int, (x : int)::xs) = if x < sum then 1 +
number_before_reaching_sum(sum, [x + (hd xs)] @ tl xs) else 0;
(* 9 *)
val days_of_months = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
fun what_month (day_of_year : int) = 1 + number_before_reaching_sum(day_of_year,
days_of_months);
(* 10 *)
fun month_range (day1 : int, day2 : int) = if day1 > day2 then [] else [day1] @
month_range(day1 + 1, day2);
(* 11 *)
fun legacy ([]) = NONE |
legacy ((x : int*int*int)::xs) = if xs = [] then SOME(x) else (if is_older(x, hd xs) then
legacy([x] @ tl xs) else legacy(xs));
(* val res = legacy ([(1999, 12, 11), (2003, 11, 29), (2001, 5, 14)]); *)

(* TESTS *)
fun assertEqual (a1, a2) = if a1 = a2 then "True" else "False";
print("Test1: is_older:\n");
print("1) " ^ assertEqual(is_older((2001, 11, 5), (2002, 10, 2)), true) ^ "\n");
```

```
print("2) " ^ assertEqual(is_older((2002, 10, 2), (2002, 10, 2)), false) ^ "\n");
print("3) " ^ assertEqual(is_older((2003, 10, 2), (2002, 10, 2)), false) ^ "\n");

print("Test2: number_in_month:\n");
print("1) " ^ assertEqual(number_in_month([(2003, 11, 8), (2010, 5, 26)], 5, 0), 1) ^
"\n");
print("2) " ^ assertEqual(number_in_month([(2003, 5, 8), (2010, 5, 26)], 5, 0), 2) ^
"\n");
print("3) " ^ assertEqual(number_in_month([(2003, 1, 8), (2010, 2, 26)], 5, 0), 0) ^
"\n");

print("Test3: number_in_months:\n");
print("1) " ^ assertEqual(number_in_months([(2003, 11, 8), (2010, 5, 26)], [5, 11], 0), 2)
^ "\n");
print("2) " ^ assertEqual(number_in_months([(2003, 5, 8), (2010, 3, 26)], [5], 0), 1) ^
"\n");
print("3) " ^ assertEqual(number_in_months([(2003, 1, 8), (2010, 2, 26)], [3], 0), 0) ^
"\n");

print("Test4: dates_in_month:\n");
print("1) " ^ assertEqual(dates_in_month([(2003, 5, 8), (2010, 5, 26)], 5), [(2003, 5, 8),
(2010, 5, 26)]) ^ "\n");
print("2) " ^ assertEqual(dates_in_month([(2003, 5, 8), (2010, 3, 26)], 5), [(2003, 5,
8)]) ^ "\n");
print("3) " ^ assertEqual(dates_in_month([(2003, 1, 8), (2010, 2, 26)], 3), []) ^ "\n");

print("Test5: dates_in_months:\n");
print("1) " ^ assertEqual(dates_in_months([(2003, 5, 8), (2010, 8, 26)], [5, 8]), [(2003,
5, 8), (2010, 8, 26)]) ^ "\n");
print("2) " ^ assertEqual(dates_in_months([(2003, 5, 8), (2010, 3, 26)], [5]), [(2003, 5,
8)]) ^ "\n");
print("3) " ^ assertEqual(dates_in_months([(2003, 1, 8), (2010, 2, 26)], [3]), []) ^
"\n");

print("Test6: get_nth:\n");
print("1) " ^ assertEqual(get_nth(months_names, 1), "January") ^ "\n");
print("2) " ^ assertEqual(get_nth(months_names, 3), "March") ^ "\n");
print("3) " ^ assertEqual(get_nth(months_names, 8), "August") ^ "\n");

print("Test7: date_to_string:\n");
print("1) " ^ assertEqual(date_to_string((2002, 10, 08)), "October 8, 2002") ^ "\n");
print("2) " ^ assertEqual(date_to_string((2003, 12, 11)), "December 11, 2003") ^ "\n");

print("Test8: number_before_reaching_sum:\n");
print("1) " ^ assertEqual(number_before_reaching_sum(10, [1, 1, 2, 1, 2, 2, 1]), 6) ^
"\n");
print("2) " ^ assertEqual(number_before_reaching_sum(5, [1, 1, 2, 1, 2, 2, 1]), 3) ^
"\n");
print("3) " ^ assertEqual(number_before_reaching_sum(2, [3, 1, 2, 1, 2, 2, 1]), 0) ^
"\n");
```

```
print("Test9: what_month:\n");
print("1) " ^ assertEqual(what_month(334), 11) ^ "\n");
print("2) " ^ assertEqual(what_month(335), 12) ^ "\n");
print("3) " ^ assertEqual(what_month(32), 2) ^ "\n");

print("Test10: month_range:\n");
print("1) " ^ assertEqual(month_range(8, 19), [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19]) ^ "\n");
print("2) " ^ assertEqual(month_range(1, 2), [1, 2]) ^ "\n");

print("Test11: legacy:\n");
print("1) " ^ assertEqual(legacy([(1999, 12, 11), (2003, 11, 29), (2001, 5, 14)]),
SOME((1999, 12, 11))) ^ "\n");
print("2) " ^ assertEqual(legacy([]), NONE) ^ "\n");
```

4. Скріншоти виконання коду

```
Standard ML of New Jersey (32-bit) v110.99.2 [built: Tue Sep 28 13:04:14 2021]
- use "lab2.sml";
[opening lab2.sml]
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val number_in_month = fn : (int * int * int) list * int * int -> int
val number_in_months = fn : (int * int * int) list * int list * int -> int
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list
val dates_in_months = fn :
  (int * int * int) list * int list -> (int * int * int) list
val get_nth = fn : string list * int -> string
val months_names =
  ["January","February","March","April","May","June","July","August",
   "September","October","November","December"] : string list
[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 : int * int * int -> string
lab2.sml:25.5-25.142 Warning: match nonexhaustive
          (sum,x :: xs) => ...

val number_before_reaching_sum = fn : int * int list -> int
val days_of_months = [31,28,31,30,31,30,31,31,30,31,30,31] : int list
val what_month = fn : int -> int
val month_range = fn : int * int -> int list
val legacy = fn : (int * int * int) list -> (int * int * int) option
lab2.sml:37.34 Warning: calling polyEqual
val assertEqual = fn : ''a * ''a -> string
```

```
Test1: is_older:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test2: number_in_month:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test3: number_in_months:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test4: dates_in_month:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test5: dates_in_months:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
```

```
Test6: get_nth:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test7: date_to_string:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
Test8: number_before_reaching_sum:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test9: what_month:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
3) True
val it = () : unit
Test10: month_range:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
Test11: legacy:
val it = () : unit
1) True
val it = () : unit
2) True
val it = () : unit
val it = () : unit
```