**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)**

(шифр, прізвище, ім'я, по батькові)

*ІП-01 Пашковський Євгеній Сергійович*

**Перевірив**

(прізвище, ім'я, по батькові)

*ас. Очеретяний О.К.*

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, ..., 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.
12. Опис використаних технологій

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

1. 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");

1. Скріншоти виконання коду



