Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

ЗВІТ

до лабораторних робіт №2

Мультипарадигменне програмування

Виконав студент ІПз01 Максиміхін О.В.
(№ групи, прізвище, ім’я, по батькові)

Прийняв ас. Очеретяний О. К.
(посада, прізвище, ім’я, по батькові)

Київ 2022

Завдання.

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.

Програмний код

lab2func.sml

```
(*Task_01*)
fun is_older (date_1: int* int* int, date_2: int* int* int) =
    if (#1 date_1) < (#1 date_2) then
        true
    else if (#1 date_1) = (#1 date_2) then
            if (#2 date_1) < (#2 date_2) then
                true
            else if ((#2 date_1) = (#2 date_2)) andalso ((#3 date_1) < (#3 date_2))
then
                true
                else
                    false
        else
            false;


(*Task_02*)
fun number_in_month (dates: (int* int* int) list, month: int) =
    if null dates then
        0
    else
        if #2 (hd dates) = month then
            1 + number_in_month(tl dates, month)
        else
            number_in_month(tl dates, month);

(*Task_03*)
fun number_in_months (dates: (int* int* int) list, months: int list ) =
    if null months then
        0
    else
        number_in_month(dates, hd months) + number_in_months(dates, tl months);

(*Task_04*)
fun dates_in_month (dates: (int* int* int) list, month: int) =
    if null dates then
        []
    else
        if #2 (hd dates) = month then
            (hd dates) :: dates_in_month(tl dates, month)
        else
            dates_in_month(tl dates, month);

(*Task_05*)
fun dates_in_months (dates: (int* int* int) list, months: int list) =
```

```
    if null months then
        []
    else
        dates_in_month(dates, hd months) @ dates_in_months(dates, tl months);


(*Task_06*)
fun get_nth (strings : string list, n : int) =
    if null strings then
        ""
    else
        if n = 1 then
            hd strings
        else
            get_nth(tl strings, n - 1);

(*Task_07*)
fun date_to_string (date: int* int* int) =
    get_nth(["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"]
    , #2 date) ^ " " ^ (Int.toString (#3 date)) ^ ", " ^ (Int.toString (#1 date));

(*Task_08*)
fun number_before_reaching_sum (sum: int, numbers: int list) =
    if null numbers then
        0
    else
        if (sum - (hd numbers)) <= 0 then
            0
        else
            1 + number_before_reaching_sum(sum - (hd numbers), tl numbers);

(*Task_09*)
fun what_month (dayOfYear: int) =
    number_before_reaching_sum(dayOfYear, [31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31]) + 1;

(*Task_10*)
fun month_range (dayStart: int, dayEnd: int) =
    if dayStart > dayEnd then
        []
    else
        what_month(dayStart) :: month_range(dayStart + 1, dayEnd);

(*Task_11*)
fun min_date (dates: (int* int* int) list) =
    if null dates then
```

```
    NONE
else
    let fun min_date_nonempty (dates : (int*int*int) list) =
            if null (tl dates)
            then hd dates
            else
                let val tl_ans = min_date_nonempty(tl dates)
                in
                    if is_older(hd dates, tl_ans) then
                        hd dates
                    else
                        tl_ans
                end
    in
        SOME (min_date_nonempty(dates))
    end;
```

Тести

lab2test.sml

```
use "lab2func.sml";
(*Test_Function*)
fun test(fun_name : string, result_true, result_actual) =
if result_true = result_actual
then (fun_name, "Passed")
else (fun_name, "Failed");

(*Test_Task_01*)
print "Test_Task_01:\n";
test("is_older", true, is_older((2000, 1, 30), (2021, 1, 30)));
test("is_older", true, is_older((2000, 1, 30), (2000, 1, 31)));
test("is_older", false, is_older((2000, 1, 30), (2000, 1, 30)));

(*Test_Task_02*)
print "Test_Task_02:\n";
test("number_in_month", 3, number_in_month([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], 1));
test("number_in_month", 0, number_in_month([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], 3));
test("number_in_month", 1, number_in_month([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], 12));

(*Test_Task_03*)
print "Test_Task_03:\n";
test("number_in_months", 5, number_in_months([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], [1, 2]));
test("number_in_months", 3, number_in_months([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], [1, 3]));
test("number_in_months", 0, number_in_months([(2022, 1, 1), (2021, 1, 1), (2020, 1,
30), (2021, 2, 31), (2020, 12, 31), (2020, 2, 1)], [3, 10, 11]));

(*Test_Task_04*)
print "Test_Task_04:\n";
test("dates_in_month", [(1981, 3, 3), (1981, 3, 10)], dates_in_month([(1981, 3, 3),
(2001, 1, 3), (1981, 3, 10), (2005, 6, 30), (2022, 7, 30)], 3));
test("dates_in_month", [(1981, 3, 3)], dates_in_month([(1981, 3, 3), (2001, 1, 3),
(1981, 4, 10), (2005, 6, 30), (2022, 7, 30)], 3));
test("dates_in_month", [], dates_in_month([(1981, 3, 3), (2001, 1, 3), (1981, 3,
10), (2005, 6, 30), (2022, 7, 30)], 10));

(*Test_Task_05*)
print "Test_Task_05:\n";
```

```
test("dates_in_months", [(1981, 3, 3), (1981, 3, 10), (2022, 7, 30)],
dates_in_months([(1981, 3, 3), (2001, 1, 3), (1981, 3, 10), (2005, 6, 30), (2022,
7, 30)], [3, 7]));
test("dates_in_months", [(2001, 1, 3)], dates_in_months([(1981, 3, 3), (2001, 1,
3), (1981, 3, 10), (2005, 6, 30), (2022, 7, 30)], [1, 2]));
test("dates_in_months", [], dates_in_months([(1981, 3, 3), (2001, 1, 3), (1981, 3,
10), (2005, 6, 30), (2022, 7, 30)], [8, 9]));

(*Test_Task_06*)
print "Test_Task_06:\n";
test("get_nth", "apple", get_nth(["apple", "ball", "cat", "door"], 1));
test("get_nth", "door", get_nth(["apple", "ball", "cat", "door"], 4));
test("get_nth", "", get_nth(["apple", "ball", "cat", "door"], 5));

(*Test_Task_07*)
print "Test_Task_07:\n";
test("date_to_string", "March 3, 1981", date_to_string((1981, 3, 3)));
test("date_to_string", "January 2, 1948", date_to_string((1948, 1, 2)));
test("date_to_string", "April 18, 1950", date_to_string((1950, 4, 18)));

(*Test_Task_08*)
print "Test_Task_08:\n";
test("number_before_reaching_sum", 0, number_before_reaching_sum(6, [6, 5, 4, 3, 2,
1]));
test("number_before_reaching_sum", 2, number_before_reaching_sum(15, [6, 5, 4, 3,
2, 1]));
test("number_before_reaching_sum", 6, number_before_reaching_sum(22, [6, 5, 4, 3,
2, 1]));

(*Test_Task_09*)
print "Test_Task_09:\n";
test("what_month", 1, what_month(30));
test("what_month", 1, what_month(31));
test("what_month", 2, what_month(32));

(*Test_Task_10*)
print "Test_Task_10:\n";
test("month_range", [], month_range(31, 30));
test("month_range", [1], month_range(31, 31));
test("month_range", [1, 2], month_range(31, 32));
test("month_range", [1, 2, 2], month_range(31, 33));

(*Test_Task_11*)
print "Test_Task_10:\n";
test("min_date", NONE, min_date([]));
test("min_date", SOME (1948, 1, 2), min_date([(1948, 1, 2), (1950, 4, 18), (1981,
3, 3), (2010, 6, 25)]));
```

```
test("min_date", SOME (1948, 1, 2), min_date([(1950, 4, 18), (1981, 3, 3), (2010,
6, 25), (1948, 1, 2)]));
```

Результати виконання тестів

```
Test_Task_01:
val it = () : unit
val it = ("is_older","Passed") : string * string
val it = ("is_older","Passed") : string * string
val it = ("is_older","Passed") : string * string
Test_Task_02:
val it = () : unit
val it = ("number_in_month","Passed") : string * string
val it = ("number_in_month","Passed") : string * string
val it = ("number_in_month","Passed") : string * string
Test_Task_03:
val it = () : unit
val it = ("number_in_months","Passed") : string * string
val it = ("number_in_months","Passed") : string * string
val it = ("number_in_months","Passed") : string * string
Test_Task_04:
val it = () : unit
val it = ("dates_in_month","Passed") : string * string
val it = ("dates_in_month","Passed") : string * string
val it = ("dates_in_month","Passed") : string * string
Test_Task_05:
val it = () : unit
val it = ("dates_in_months","Passed") : string * string
val it = ("dates_in_months","Passed") : string * string
val it = ("dates_in_months","Passed") : string * string
Test_Task_06:
val it = () : unit
val it = ("get_nth","Passed") : string * string
val it = ("get_nth","Passed") : string * string
val it = ("get_nth","Passed") : string * string
Test_Task_07:
val it = () : unit
val it = ("date_to_string","Passed") : string * string
val it = ("date_to_string","Passed") : string * string
val it = ("date_to_string","Passed") : string * string
Test_Task_08:
val it = () : unit
val it = ("number_before_reaching_sum","Passed") : string * string
val it = ("number_before_reaching_sum","Passed") : string * string
val it = ("number_before_reaching_sum","Passed") : string * string
```

Test_Task_09:
val it = () : unit
val it = ("what_month","Passed") : string * string
val it = ("what_month","Passed") : string * string
val it = ("what_month","Passed") : string * string
Test_Task_10:
val it = () : unit
val it = ("month_range","Passed") : string * string
val it = ("month_range","Passed") : string * string
val it = ("month_range","Passed") : string * string
val it = ("month_range","Passed") : string * string
Test_Task_11:
val it = () : unit
val it = ("min_date","Passed") : string * string
val it = ("min_date","Passed") : string * string
val it = ("min_date","Passed") : string * string