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 студент: ІП-01 Танасієнко Олександр

Перевірив: ас. Очеретяний О. К.

Київ 2022

1. **Завдання лабораторної роботи**

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.

2. Програмний код

task1.sml

```
(*task 1*)
fun is_older (x: int*int*int, y: int*int*int) =
    if (#1 x) <= (#1 y) andalso (#2 x) <= (#2 y) andalso (#3 x) < (#3
y)
    then true
    else false;



(*tests for task 1*)
fun provided_test1 () =
    let val date1 = (2022,6,15)
        val date2 = (2022,6,21)
    in
        is_older(date1,date2)
    end;

fun provided_test2 () =
    let val date1 = (2022,6,15)
        val date2 = (2022,5,15)
    in
        is_older(date1,date2)
    end;

fun provided_test3 () =
    let val date1 = (2001,3,3)
        val date2 = (2025,10,5)
    in
```

```
        is_older(date1,date2)
    end;


fun provided_test4 () =
    let val date1 = (2025,10,5)
        val date2 = (2001,3,3)
    in
        is_older(date1,date2)
    end;


val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task2.sml

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



(*tests for task 2*)
fun provided_test1 () =
    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]
```

```
        val month = 3
    in
        number_in_month(datesList, month)
    end;

fun provided_test2 () =
    let val datesList = [(2025,3,5), (2025,3,5), (2013,3,10)]
        val month = 6
    in
        number_in_month(datesList, month)
    end;

fun provided_test3 () =
    let val datesList = [(2010,6,7)]
        val month = 6
    in
        number_in_month(datesList, month)
    end;

fun provided_test4 () =
    let val datesList = [(2025,8,5), (2026,8,5)]
        val month = 8
    in
        number_in_month(datesList, month)
    end;

val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
```

```
val ans4 = provided_test4();
```

task3.sml

```
(*task 3*)

use "task2.sml";


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);




(*tests for task 3*)

fun provided_test1 () =

    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]

        val months = [3, 10]

    in

        number_in_months(datesList, months)

    end;


fun provided_test2 () =

    let val datesList = [(2025,3,5), (2025,4,5), (2013,3,10)]

        val months = [6, 7]

    in

        number_in_months(datesList, months)

    end;


fun provided_test3 () =

    let val datesList = [(2010,6,7), (2010,9,7), (2010,11,7)]
```

```
        val months = [6, 9, 11]
    in
        number_in_months(datesList, months)
    end;

fun provided_test4 () =
    let val datesList = [(2025,8,5), (2025,9,5), (2013,10,10)]
        val months = [8, 9]
    in
        number_in_months(datesList, months)
    end;

val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task4.sml

```
(*task 4*)
fun dates_in_month(dates: (int*int*int) list, month: int) =
    if (null dates)
    then []
    else ((
        if (#2 (hd dates)) = month
        then [hd dates]
        else []
    ) @ dates_in_month(tl dates, month));



(*tests for task 4*)
```

```
fun provided_test1 () =
    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]
        val month = 3
    in
        dates_in_month(datesList, month)
    end;

fun provided_test2 () =
    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]
        val month = 6
    in
        dates_in_month(datesList, month)
    end;

fun provided_test3 () =
    let val datesList = [(2010,6,7), (2010,9,7), (2010,11,7)]
        val month = 9
    in
        dates_in_month(datesList, month)
    end;

fun provided_test4 () =
    let val datesList = [(2025,8,5), (2025,8,15), (2013,8,10)]
        val month = 8
    in
        dates_in_month(datesList, month)
    end;

val ans1 = provided_test1();
```

```
val ans2 = provided_test2();

val ans3 = provided_test3();

val ans4 = provided_test4();
```

task5.sml

```
(*task 5*)

use "task4.sml";


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)

    );




(*tests for task 5*)

fun provided_test1 () =

    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]

        val months = [3, 10]

    in

        dates_in_months(datesList, months)

    end;


fun provided_test2 () =

    let val datesList = [(2025,3,5), (2025,10,5), (2013,3,10)]

        val months = [6, 7]

    in

        dates_in_months(datesList, months)
```

```
    end;


fun provided_test3 () =
    let val datesList = [(2010,6,7), (2010,9,7), (2010,11,7)]
        val months = [6, 9, 11]
    in
        dates_in_months(datesList, months)
    end;


fun provided_test4 () =
    let val datesList = [(2025,8,5), (2025,9,5), (2025,10,5)]
        val months = [8, 9]
    in
        dates_in_months(datesList, months)
    end;


val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task6.sml

```
(*task 6*)
fun get_nth(strings: string list, n: int) =
    if (n = 1)
    then hd strings
    else get_nth(tl strings, n-1);



(*tests for task 6*)
```

```
fun provided_test1 () =
    let val list = ["str1", "str2", "str3"]
        val n = 1
    in
        get_nth(list, n)
    end;

fun provided_test2 () =
    let val list = ["str1", "str2", "str3"]
        val n = 3
    in
        get_nth(list, n)
    end;

fun provided_test3 () =
    let val list = ["str1", "str2", "str3"]
        val n = 2
    in
        get_nth(list, n)
    end;

fun provided_test4 () =
    let val list = ["str1", "str2", "str3", "str4", "str5"]
        val n = 4
    in
        get_nth(list, n)
    end;

val ans1 = provided_test1();
```

```
val ans2 = provided_test2();

val ans3 = provided_test3();

val ans4 = provided_test4();
```

task7.sml

```
(*task 7*)

use "task6.sml";


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));



(*tests for task 7*)

fun provided_test1 () =

    let val date = (2022, 2, 28)

    in

        date_to_string(date)

    end;
```

```
fun provided_test2 () =
    let val date = (1995, 12, 13)
    in
        date_to_string(date)
    end;

fun provided_test3 () =
    let val date = (2022, 10, 17)
    in
        date_to_string(date)
    end;

fun provided_test4 () =
    let val date = (2007, 1, 1)
    in
        date_to_string(date)
    end;

val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task8.sml

```
(*task 8*)
fun number_before_reaching_sum(sum: int, numbers: int list) =
    let
        fun sumListNElems(numberList: int list, n: int) =
            if (n = 0)
```

```
            then 0
            else (hd numberList) + sumListNElems(tl numberList, n - 1)
        fun getIndexN(sum: int, numberList: int list, n: int) =
            if (sumListNElems(numberList, n + 1) >= sum)
            then n
            else getIndexN(sum, numberList, n + 1)
    in
        getIndexN(sum, numbers, 0)
    end;



(*tests for task 8*)
fun provided_test1 () =
    let val sum = 31
        val list = [10, 10, 10, 10]
    in
        number_before_reaching_sum(sum, list)
    end;

fun provided_test2 () =
    let val sum = 11
        val list = [10, 10, 10, 10]
    in
        number_before_reaching_sum(sum, list)
    end;

fun provided_test3 () =
    let val sum = 7
        val list = [1, 2, 3, 4, 5]
```

```
    in
        number_before_reaching_sum(sum, list)
    end;

fun provided_test4 () =
    let val sum = 55
        val list = [10, 15, 20, 10]
    in
        number_before_reaching_sum(sum, list)
    end;

val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task9.sml

```
(*task 9*)
use "task8.sml";

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



(*tests for task 9*)
fun provided_test1 () =
    let val day = 365
    in
        what_month(day)
```

```
    end;


fun provided_test2 () =
    let val day = 1
    in
        what_month(day)
    end;


fun provided_test3 () =
    let val day = 40
    in
        what_month(day)
    end;


fun provided_test4 () =
    let val day = 320
    in
        what_month(day)
    end;


val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task10.sml

```
(*task 10*)
use "task9.sml";


fun month_range(day1: int, day2: int) =
```

```
    if day1<=day2 then what_month(day1) :: month_range(day1 + 1, day2)
    else [];



(*tests for task 10*)
fun provided_test1 () =
    let val day1 = 25
        val day2 = 35
    in
        month_range(day1, day2)
    end;

fun provided_test2 () =
    let val day1 = 365
        val day2 = 1
    in
        month_range(day1, day2)
    end;

fun provided_test3 () =
    let val day1 = 1
        val day2 = 5
    in
        month_range(day1, day2)
    end;

fun provided_test4 () =
    let val day1 = 55
        val day2 = 70
```

```
    in
        month_range(day1, day2)
    end;

val ans1 = provided_test1();
val ans2 = provided_test2();
val ans3 = provided_test3();
val ans4 = provided_test4();
```

task11.sml

```
(*task 11*)
use "task1.sml";

fun legacy(dates: (int*int*int) list) =
    let
        fun getOldestDate(dates: (int*int*int) list, dateToCompare:
int*int*int) =
            if (null dates)
            then dateToCompare
            else (
                getOldestDate((tl dates), (
                    if (is_older((hd dates), dateToCompare))
                    then (hd dates)
                    else dateToCompare
                ))
            )
    in
        if null dates
        then NONE
        else SOME (getOldestDate((tl dates), (hd dates)))
```

```
    end;



(*tests for task 11*)
fun provided_test1 () =
    let val dates = [(2,2,2), (3,3,3), (4,4,4), (5,5,5)]
    in
        legacy(dates)
    end;

fun provided_test2 () =
    let val dates = []
    in
        legacy(dates)
    end;

fun provided_test3 () =
    let val dates = [(2,2,2)]
    in
        legacy(dates)
    end;

fun provided_test4 () =
    let val dates = [(3,3,3), (3,3,3)]
    in
        legacy(dates)
    end;

val ans1 = provided_test1();
```

```
val ans2 = provided_test2();

val ans3 = provided_test3();

val ans4 = provided_test4();
```

3. **Результати виконання тестів**

task 1:

```
val ans1 = true : bool

val ans2 = false : bool

val ans3 = true : bool

val ans4 = false : bool
-
```

task 2:

```
val ans1 = 2 : int

val ans2 = 0 : int

val ans3 = 1 : int

val ans4 = 2 : int
-
```

task 3:

```
val ans1 = 3 : int

val ans2 = 0 : int

val ans3 = 3 : int

val ans4 = 2 : int
-
```

task 4:

```
val ans1 = [(2025,3,5),(2013,3,10)] : (int * int * int) list

val ans2 = [] : (int * int * int) list

val ans3 = [(2010,9,7)] : (int * int * int) list

val ans4 = [(2025,8,5),(2025,8,15),(2013,8,10)] : (int * int * int) list
-
```

task 5:

```
val ans1 = [(2025,3,5),(2013,3,10),(2025,10,5)] : (int * int * int) list

val ans2 = [] : (int * int * int) list

val ans3 = [(2010,6,7),(2010,9,7),(2010,11,7)] : (int * int * int) list

val ans4 = [(2025,8,5),(2025,9,5)] : (int * int * int) list
-
```

task 6:

```
val ans1 = "str1" : string

val ans2 = "str3" : string

val ans3 = "str2" : string

val ans4 = "str4" : string
-
```

task 7:

```
val ans1 = "February 28, 2022" : string

val ans2 = "December 13, 1995" : string

val ans3 = "October 17, 2022" : string

val ans4 = "January 1, 2007" : string
-
```

task 8:

```
val ans1 = 3 : int

val ans2 = 1 : int

val ans3 = 3 : int

val ans4 = 3 : int
-
```

task 9:

```
val ans1 = 12 : int

val ans2 = 1 : int

val ans3 = 2 : int

val ans4 = 11 : int
-
```

task 10:

```
val ans1 = [1,1,1,1,1,1,1,2,2,2,2] : int list

val ans2 = [] : int list

val ans3 = [1,1,1,1,1] : int list

val ans4 = [2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3] : int list
-
```

task 11:

```
val ans1 = SOME (2,2,2) : (int * int * int) option

val ans2 = NONE : (int * int * int) option

val ans3 = SOME (2,2,2) : (int * int * int) option

val ans4 = SOME (3,3,3) : (int * int * int) option
```