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

Кафедра інформатики та програмної інженерії

Програмування інтелектуальних інформаційних систем

ЗВІТ

до лабораторних робіт

Виконав студент ІП-01 Смислов Даніл _____
(№ групи, прізвище, ім'я, по батькові)

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

Київ 2022

1. Завдання лабораторної роботи

Написати 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.

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. Опис програмного коду

Функція 1:

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

Функція 2:

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

Функція 3:

```
(*3*)
fun number_in_months(dateList: (int*int*int) list, monthList: int list) =
if null monthList orelse null dateList
then 0
else
    if number_in_month(dateList, hd monthList) <> 0
    then number_in_months(dateList, tl monthList) + number_in_month(dateList, hd
monthList)
    else
        number_in_months(dateList,tl monthList);
```

Функція 4:

```
(*4*)
fun dates_in_month(dateList: (int*int*int) list, monthNumber: int) =
if null dateList
then []
else
    if (#2 (hd dateList)) = monthNumber
```

```
        then (hd dateList) :: dates_in_month(tl dateList,monthNumber)
    else dates_in_month(tl dateList,monthNumber)
```

Функція 5:

```
(*5*)
fun dates_in_months(dateList: (int*int*int) list, monthList: int list) =
if null monthList
then []
else
    dates_in_month(dateList,hd monthList) @ dates_in_months(dateList,tl
monthList)
```

Функція 6:

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

Функція 7:

```
(*7*)
fun date_to_string(date:int*int*int) =
let val lst =
["January","February","March","April","May","June","July","August","September","O
ctober","November","December"]
in
get_nth(lst,#2 date) ^ " " ^(Int.toString (#3 date)) ^ ", " ^ (Int.toString (#1
date))
end;
```

Функція 8:

```
(*8*)
fun number_before_reaching_sum(sum:int, lst: int list)=
if null lst
then 0
```

```
else
if(sum - hd lst > 0)
then
number_before_reaching_sum(sum-hd lst, tl lst) + 1
else
0;
```

Функція 9:

```
(*9*)
fun what_month(day:int)=
if day > 0 andalso day < 366
then
let val lst = [31,28,31,30,31,30,31,31,30,31,30,31]
in
number_before_reaching_sum(day,lst) + 1
end
else
~1;
```

Функція 10:

```
(*10*)
fun month_range(day1:int,day2:int) =
if day1 > day2
then []
else
what_month(day1)::month_range(day1+1,day2);
```

Функція 11:

```
(*11*)

fun what_oldest_date(lst: (int*int*int) list) =
if null lst
then NONE
else
let
    fun oldest_nonempty(lst: (int*int*int) list)=
    if null(tl lst)
    then hd lst
    else
        let val tl_ans = oldest_nonempty(tl lst)
        in
            if is_older(hd lst, tl_ans)
            then hd lst
            else tl_ans
        end
in
```

```
    SOME (oldest_nonempty lst)
end;
```

3. Скріншоти роботи функцій

Тестування функції 1:

```
fun test1() =
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2021,01,20)
in
(is_older(date1,date2),
is_older(date1,date3))
end

val ans1 = test1() (*expected: (true,false) *)
```

Результат тестування функції 1:

```
val is_older = fn : (int * int * int) * (int * int * int) -> bool

val test1 = fn : unit -> bool * bool
val ans1 = (true,false) : bool * bool
```

Тестування функції 2:

```
fun test2() =
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2023,01,25)
    val dataList = [date1,date2,date3]
    val monthNumber = 2;
in
(number_in_month(dataList,1),
number_in_month(dataList,3))
end

val ans2 = test2() (*expected: (2,0)*)
```

Результат тестування функції 2:

```
val number_in_month = fn : (int * int * int) list * int -> int

val test2 = fn : unit -> int * int
val ans2 = (2,0) : int * int
```

Тестування функції 3:

```
fun test3()=
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2023,01,25)
    val date4 = (1999,9,24)
    val dateList = [date1,date2,date3]
    val dateList2 = [date1,date2,date2,date4]
    val monthList = [1,2,3]
    val monthList2 = [1,9]
in
(number_in_months(dateList,monthList),
number_in_months(dateList2,monthList2))
end

val ans3 = test3(); (*expected: (3,2) *)
```

Результат тестування функції 3:

```
val number_in_months = fn : (int * int * int) list * int list -> int

val test3 = fn : unit -> int * int
val ans3 = (3,2) : int * int
```

Тестування функції 4:

```
fun test4()=
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2023,03,25)
    val date4 = (2019,01,29)
    val dateList = [date1,date2,date3,date4]
in
(dates_in_month(dateList,1),
dates_in_month(dateList,4))
end

val ans4 = test4(); (*expected: ([(2021,1,20),(2019,1,29)],[]) *)
```

Результат тестування функції 4:

```
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list
val test4 = fn : unit -> (int * int * int) list * (int * int * int) list
val ans4 = ([(2021,1,20),(2019,1,29)],[]) :
  (int * int * int) list * (int * int * int) list
```

Тестування функції 5:

```
fun test5()=
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2023,03,25)
    val date4 = (2019,04,29)
    val dateList = [date1,date2,date3,date4]
    val dateList2 = [date1,date4,date4]
    val monthList = [1,2,3]
in
(dates_in_months(dateList,monthList),
dates_in_months(dateList2,monthList))
end

val ans5 =  test5(); (*expected: ([(2021,01,20),(2021,02,20),(2023,03,25)],
[(2021,01,20)])*)
```

Результат тестування функції 5:

```
val dates_in_months = fn :
  (int * int * int) list * int list -> (int * int * int) list
val test5 = fn : unit -> (int * int * int) list * (int * int * int) list
val ans5 = ([(2021,1,20),(2021,2,20),(2023,3,25)],[(2021,1,20)]) :
  (int * int * int) list * (int * int * int) list
```

Тестування функції 6:

```
fun test6()=
let val str1 = "check1"
    val str2 = "check2"
    val str3 = "check3"
    val lst = [str1,str2,str3]
in
(get_nth(lst,2),
get_nth(lst,4))
end

val ans6 = test6(); (*expected: ("check2","")*)
```

Результат тестування функції 6:

```
val get_nth = fn : string list * int -> string

val test6 = fn : unit -> string * string
val ans6 = ("check2","") : string * string
```

Тестування функції 7:

```
fun test7()=
let val date1 = (1991,8,24)
    val date2 = (2003,9,24)
in
(date_to_string(date1),
date_to_string(date2))
end

val ans7 = test7();(*expected: ("August 24, 1991","September 24, 2003")*)
```

Результат тестування функції 7:

```
val date_to_string = fn : int * int * int -> string

val test7 = fn : unit -> string * string
val ans7 = ("August 24, 1991","September 24, 2003") : string * string
```

Тестування функції 8:

```
fun test8()=
let val lst = [1,2,3,4,5]
    val sum = 15
    val lst2 = [10,5,1,5]
    val sum2 = 16
in
(number_before_reaching_sum(sum,lst),
number_before_reaching_sum(sum2,lst2))
end

val ans8 = test8(); (*expected: (4,2)*)
```

Результат тестування функції 8:

```
val number_before_reaching_sum = fn : int * int list -> int

val test8 = fn : unit -> int * int
val ans8 = (4,2) : int * int
```

Тестування функції 9:

```
    fun test9()=

let val day1 = 32
    val day2 = 91
    val day3 = 365
in
(what_month(day1),
what_month(day2),
what_month(day3))
```

```
end

val ans9 = test9(); (*expected: (2,4,12)*)
```

Результат тестування функції 9:

```
val what_month = fn : int -> int

val test9 = fn : unit -> int * int * int
val ans9 = (2,4,12) : int * int * int
```

Тестування функції 10:

```
fun test10()=
let val day1 = 80
    val day2 = 95
    val day3 = 230
    val day4 = 130
in
(month_range(day1,day2),
month_range(day3,day4))
end

val ans10 = test10(); (*expected: ([3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4],[]) *)
```

Результат тестування функції 10:

```
val month_range = fn : int * int -> int list

val test10 = fn : unit -> int list * int list
val ans10 = ([3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4],[]) : int list * int list
```

Тестування функції 11:

```
fun test11()=
let val date1 = (2021,01,20)
    val date2 = (2021,02,20)
    val date3 = (2019,04,29)
    val date4 = (2023,03,25)
    val dateList = [date1,date2,date3,date4]
    val dateList2 = []
in
(what_oldest_date(dateList),
what_oldest_date(dateList2))
end
```

```
val ans11 = test11(); (*expected: (SOME (2019,4,29),NONE) *)
```

Результат тестування функції 11:

```
val what_oldest_date = fn : (int * int * int) list -> (int * int * int) option

val test11 = fn : unit -> (int * int * int) option * (int * int * int) option
val ans11 = (SOME (2019,4,29),NONE) :
  (int * int * int) option * (int * int * int) option
```