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 *ІП-01 Заранік Богдан* _____
(шифр, прізвище, ім'я, по батькові)

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

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, ..., 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

Для виконання даної лабораторної роботи мною було використано мову програмування SML, оскільки такою була вимога завдання. У якості середовіща розробки - VSCode та SMLCompiler.

3. ОПИС ПРОГРАМНОГО КОДУ

Task1

```
(*Task 1*)
fun is_older(firstDate: int*int*int, secondDate: int*int*int) =
    if (#1 firstDate) < (#1 secondDate) then true
    else if (#1 firstDate) > (#1 secondDate) then false
    (*here year1 == year2*)
    else
        if (#2 firstDate) < (#2 secondDate) then true
        else if (#2 firstDate) > (#2 secondDate) then false
        else
            if (#3 firstDate) < (#3 secondDate) then true
            else if (#3 firstDate) > (#3 secondDate) then false
            else false;

(* === Tests === *)
fun provided_test1 () =
    let val date1 = (2002, 7, 24)
         val date2 = (2005, 5, 11)
    in
         is_older(date1, date2)
    end;
fun provided_test2 () =
    let val date1 = (1966, 4, 21)
         val date2 = (2005, 5, 11)
    in
         is_older(date1, date2)
    end;
fun provided_test3 () =
    let val date1 = (1966, 4, 21)
         val date2 = (1966, 4, 21)
    in
         is_older(date1, date2)
    end;
fun provided_test4 () =
    let val date1 = (2002, 7, 24)
         val date2 = (1966, 4, 21)
    in
         is_older(date1, date2)
    end;
(*Tests for Task 1*)
val meAndBrother = provided_test1();
val fatherAndBrother = provided_test2();
val equalDatesTest = provided_test3();
val meAndFather = provided_test4();
```

Task2

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

(* === Tests === *)
fun provided_test1 () =
    let val array = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
    in
         number_in_month(array, 4)
    end;
fun provided_test2 () =
    let val array = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
    in
         number_in_month(array, 3)
    end;
fun provided_test3 () =
    let val array = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
    in
         number_in_month(array, 10)
    end;
val cnt = provided_test1();(*2 times here*)
val cnt = provided_test2();(*1 time here*)
val cnt = provided_test3();(*0 times here*)
```

Task3

```
(*Task3*)
fun number_in_month(dateList: (int*int*int) list, monthNumber: int) =
    if null dateList then 0
    else
        number_in_month( tl dateList, monthNumber ) +
            (if #2 (hd dateList) = monthNumber then 1 else 0);

fun solveProblem3(dateList: (int*int*int) list, monthList: int list) =
    if null monthList then 0
        else number_in_month(dateList, hd monthList)
            + solveProblem3(dateList, tl monthList);
(* === Tests === *)
fun provided_test1 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = [3, 4, 5, 7]
    in
         solveProblem3(dateList, monthList)
```

```
    end;

fun provided_test2 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = [3, 4]
    in
         solveProblem3(dateList, monthList)
    end;

fun provided_test3 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = []
    in
         solveProblem3(dateList, monthList)
    end;

val cnt = provided_test1();(*all ==> cnt = 5*)
val cnt = provided_test2();(*cnt = 3*)
val cnt = provided_test3();(*cnt = 0*)
```

Task4

```
(*Task4*)
fun dates_in_month(dateList: (int*int*int) list, month: int) =
    if null dateList then []
    else (
        if #2 (hd dateList) = month
        then (hd dateList)::dates_in_month(tl dateList, month)
        else dates_in_month(tl dateList, month)
    );


(* === Tests === *)
fun provided_test1 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val month = 4
    in
         dates_in_month(dateList, month)
    end;

fun provided_test2 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val month = 3
    in
         dates_in_month(dateList, month)
    end;
```

```
fun provided_test3 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
        (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val month = 12
    in
         dates_in_month(dateList, month)
    end;

val resultList = provided_test1();
val resultList = provided_test2();
val resultList = provided_test3();
```

Task5

```
(*Task5*)
fun number_in_month(dateList: (int*int*int) list, monthNumber: int) =
    if null dateList then []
    else
        (
            if #2 (hd dateList) = monthNumber
            then (hd dateList)::number_in_month( tl dateList, monthNumber )
            else number_in_month( tl dateList, monthNumber )
        );

fun solveProblem5(dateList: (int*int*int) list, monthList: int list) =
    if null monthList then []
    else
    number_in_month(dateList, hd monthList)@solveProblem5(dateList, tl monthList);
(* === Tests === *)
fun provided_test1 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = [3, 4, 5, 7]
    in
         solveProblem5(dateList, monthList)
    end;

fun provided_test2 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = [3, 4]
    in
         solveProblem5(dateList, monthList)
    end;

fun provided_test3 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
            (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
        val monthList = []
    in
         solveProblem5(dateList, monthList)
```

```
        end;
    val cnt = provided_test1();(*all ==> cnt = 5*)
    val cnt = provided_test2();(*cnt = 3*)
    val cnt = provided_test3();(*cnt = 0*)
```

Task6

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


    (* === Tests === *)
    fun provided_test1 () =
        let val array = ["evgeniya", "alexandra", "anna",
                "kamila", "daria", "alina"]
            val n = 1
        in
             get_nth(array, n)
        end;
    fun provided_test2 () =
        let val array = ["evgeniya", "alexandra", "anna",
                "kamila", "daria", "alina"]
            val n = 3
        in
            get_nth(array, n)
        end;
    fun provided_test3 () =
        let val array = ["evgeniya", "alexandra", "anna",
                "kamila", "daria", "alina"]
            val n = 6
        in
            get_nth(array, n)
        end;
    fun provided_test4 () =
        let val array = ["evgeniya", "alexandra", "anna",
                "kamila", "daria", "alina"]
            val n = 7
        in
            get_nth(array, n)
        end;
    fun provided_test5 () =
        let val array = []
            val n = 7
        in
            get_nth(array, n)
end;
```

```
val testCase = provided_test1();
val testCase = provided_test2();
val testCase = provided_test3();
val testCase = provided_test4();
val testCase = provided_test5();
```

Task7

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

val monthNames = [
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
];
fun date_to_string(date: (int*int*int)) =
    get_nth(monthNames, #2 date)^ " "
    ^ Int.toString(#3 date) ^ ", " ^ Int.toString(#1 date);
fun provided_test1 () =
    let val date = (1966, 04, 21)
    in
        date_to_string(date)
    end;
fun provided_test2 () =
    let val date = (2008, 08, 08)
    in
        date_to_string(date)
    end;
(* === Tests === *)
val stringDate = provided_test1();
val stringDate = provided_test2();
```

Task8

```
(*
    length of array has to be enough to
    find such value described before
*)
fun number_before_reaching_sum(sum: int, array: int list) =
    if null array then 0
    else if sum - hd(array) <= 0 then 0
    else 1 + number_before_reaching_sum(sum - hd(array), tl(array));

(* === Tests === *)
fun provided_test1 () =
    let val array = [1, 2, 3, 4, 5, 6, 7]
        val sum = 14
    in
        number_before_reaching_sum( sum, array )
    end;
fun provided_test2 () =
    let val array = [1, 2, 3, 4, 5, 6, 7]
        val sum = 15
    in
        number_before_reaching_sum( sum, array )
    end;
fun provided_test3 () =
    let val array = [1, 2, 3, 4, 5, 6, 7]
        val sum = 16
    in
        number_before_reaching_sum( sum, array )
    end;
val n = provided_test1();(*n = 4*)
val n = provided_test2();(*n = 4*)
val n = provided_test3();(*n = 5*)
```

Task9

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


fun what_month(dayOfYear: int, daysPerMonth: int list) =
    1 + number_before_reaching_sum(dayOfYear, daysPerMonth);
(* === Tests === *)
fun provided_test1 () =
    let val daysPerMonth = [31,28,31,30,31,30,31,31,30,31,30,31]
        val dayOfYear = 33
    in
        what_month(dayOfYear, daysPerMonth)
```

```
        end;
    val resultMonth = provided_test1();
```

Task10

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

    fun what_month(dayOfYear: int, daysPerMonth: int list) =
1 + number_before_reaching_sum(dayOfYear, daysPerMonth);

    fun getNthElement(array: string list, pos: int) =
        if pos = 1 then hd array
        else getNthElement(tl array, pos - 1);
    val daysPerMonth = [31,28,31,30,31,30,31,31,30,31,30,31];

    val monthNames = [
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
    ];

    fun month_range(day1: int, day2: int) =
        if day1 > day2 then []
        else getNthElement(
                monthNames,
                what_month(day1, daysPerMonth)
            )::month_range(day1+1, day2);

    (* result *)

    (* === Tests === *)

    fun provided_test1 () =
        let val day1 = 29
            val day2 = 33
        in
            month_range(day1, day2)
end;

    val result = provided_test1();
```

Task11

```
fun is_older(firstDate: int*int*int, secondDate: int*int*int) =
    if (#1 firstDate) < (#1 secondDate) then true
    else if (#1 firstDate) > (#1 secondDate) then false
    (*here year1 == year2*)
    else
        if (#2 firstDate) < (#2 secondDate) then true
        else if (#2 firstDate) > (#2 secondDate) then false
        else
            if (#3 firstDate) < (#3 secondDate) then true
            else if (#3 firstDate) > (#3 secondDate) then false
            else false;



fun legacy(xs: (int*int*int) list) =
    if null xs
    then NONE
    else
        let val tl_ans = legacy(tl xs)
        in if isSome( tl_ans ) andalso is_older( valOf( tl_ans ), hd( xs ) )
            then tl_ans
            else SOME( hd( xs ) )
        end;

(* === Tests === *)

fun provided_test1 () =
    let val dateList = [(2002, 7, 24), (2005, 5, 11),
    (1966, 4, 21), (1961, 4, 12), (1954, 3, 13)]
    in
        legacy( dateList )
end;

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

val result = provided_test1();
val result = provided_test2();
```

4. КРІНШОТИ РОБОТИ ПРОГРАМИ

Task1

```
[opening Task1.sml]
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val provided_test1 = fn : unit -> bool
val provided_test2 = fn : unit -> bool
val provided_test3 = fn : unit -> bool
val provided_test4 = fn : unit -> bool
val meAndBrother = true : bool
val fatherAndBrother = true : bool
val equalDatesTest = false : bool
val meAndFather = false : bool
```

Task2

```
val number_in_month = fn : (int * int * int) list * int -> int
val provided_test1 = fn : unit -> int
val provided_test2 = fn : unit -> int
val provided_test3 = fn : unit -> int
val cnt = 2 : int
val cnt = 1 : int
val cnt = 0 : int
```

Task3

```
val number_in_month = fn : (int * int * int) list * int -> int
val solveProblem3 = fn : (int * int * int) list * int list -> int
val provided_test1 = fn : unit -> int
val provided_test2 = fn : unit -> int
val provided_test3 = fn : unit -> int
val cnt = 5 : int
val cnt = 3 : int
val cnt = 0 : int
```

Task4

```
val dates_in_month = fn
  : (int * int * int) list * int -> (int * int * int) list
val provided_test1 = fn : unit -> (int * int * int) list
val provided_test2 = fn : unit -> (int * int * int) list
val provided_test3 = fn : unit -> (int * int * int) list
val resultList = [(1966,4,21),(1961,4,12)] : (int * int * int) list
val resultList = [(1954,3,13)] : (int * int * int) list
val resultList = [] : (int * int * int) list
```

Task5

```
val number_in_month = fn
  : (int * int * int) list * int -> (int * int * int) list
val solveProblem5 = fn
  : (int * int * int) list * int list -> (int * int * int) list
val provided_test1 = fn : unit -> (int * int * int) list
val provided_test2 = fn : unit -> (int * int * int) list
val provided_test3 = fn : unit -> (int * int * int) list
val cnt = [(1954,3,13),(1966,4,21),(1961,4,12),(2005,5,11),(2002,7,24)]
  : (int * int * int) list
val cnt = [(1954,3,13),(1966,4,21),(1961,4,12)] : (int * int * int) list
val cnt = [] : (int * int * int) list
```

Task6

```
val get_nth = fn : string list * int -> string
val provided_test1 = fn : unit -> string
val provided_test2 = fn : unit -> string
val provided_test3 = fn : unit -> string
val provided_test4 = fn : unit -> string
val provided_test5 = fn : unit -> string
val testCase = "evgeniya" : string
val testCase = "anna" : string
val testCase = "alina" : string

uncaught exception Empty
  raised at: smlnj/init/pervasive.sml:193.19-193.24
```

Task7

```
val get_nth = fn : string list * int -> string
val monthNames =
  ["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
val provided_test1 = fn : unit -> string
val provided_test2 = fn : unit -> string
val stringDate = "April 21, 1966" : string
val stringDate = "August 8, 2008" : string
```

Task8

```
val number_before_reaching_sum = fn : int * int list -> int
val provided_test1 = fn : unit -> int
val provided_test2 = fn : unit -> int
val provided_test3 = fn : unit -> int
val n = 4 : int
val n = 4 : int
val n = 5 : int
```

Task9

```
val number_before_reaching_sum = fn : int * int list -> int
val what_month = fn : int * int list -> int
val provided_test1 = fn : unit -> int
val resultMonth = 2 : int
```

Task10

```
val number_before_reaching_sum = fn : int * int list -> int
val what_month = fn : int * int list -> int
val getNthElement = fn : string list * int -> string
val daysPerMonth = [31,28,31,30,31,30,31,31,30,31,30,31] : int list
val monthNames =
  ["January","February","March","April","May","June","July","August",
   "September","October","November","December"] : string list
val month_range = fn : int * int -> string list
val provided_test1 = fn : unit -> string list
val result = ["January","January","January","February","February"]
  : string list
```

Task11

```
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val legacy = fn : (int * int * int) list -> (int * int * int) option
val provided_test1 = fn : unit -> (int * int * int) option
val provided_test2 = fn : unit -> (int * int * int) option
val result = SOME (1954,3,13) : (int * int * int) option
val result = NONE : (int * int * int) option
```