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

«Мультипарадигмене програмування»

«Фунціональне програмування»

Виконав(ла) *ІП-01, Адамчук Ілля Іванович* _____

(шифр, прізвище, ім'я, по батькові)

Перевірив Очеретяний О. К. _____

(прізвище, ім'я, по батькові)

Київ 2022

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.)

Код програми SML:

1)

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

is_older((1995,2,3),(1995,2,3)); (*false*)
is_older((1999,1,15),(1999,1,16)); (*true*)
is_older((1999,1,15),(1999,2,15)); (*true*)
is_older((1999,1,15),(2000,1,15)); (*true*)
is_older((1999,1,15),(1998,1,15)); (*false*)
is_older((1999,1,15),(1999,1,14)); (*false*)
is_older((1999,1,15),(1999,0,15)); (*false*)
```

Приклад виконання

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

2)

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

number_in_month([],4); (*0*)
number_in_month([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],4); (*3*)
number_in_month([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],3); (*1*)
number_in_month([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],7); (*0*)
number_in_month([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],5); (*1*)
```

Приклад виконання

```
val number_in_month = fn : (int * int * int) list * int -> int
val it = 0 : int
val it = 3 : int
val it = 1 : int
val it = 0 : int
val it = 1 : int
```

3)

```
fun number_in_months(list_of_dates : (int*int*int) list, monthes : int list) =
    if null monthes
        then 0
    else number_in_month(list_of_dates, (hd monthes)) +
number_in_months(list_of_dates, (tl monthes));

number_in_months([],[4,3]); (*0*)
number_in_months([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],[]); (*0*)
number_in_months([],[]); (*0*)
number_in_months([(1995,2,3), (1995,3,3), (1995,4,3), (1995,4,3), (1995,5,3),
(1995,4,3)],[4,3]); (*4*)
```

Приклад виконання

```
val number_in_months = fn : (int * int * int) list * int list -> int
val it = 0 : int
val it = 0 : int
val it = 0 : int
val it = 4 : int
```

4)

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

```
dates_in_month([],4);
dates_in_month([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],4);
dates_in_month([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],5);
dates_in_month([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],10);
```

Приклад виконання

```
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list
val it = [] : (int * int * int) list
val it = [(1995,4,1),(1995,4,2),(1995,4,3)] : (int * int * int) list
val it = [(1995,5,3)] : (int * int * int) list
val it = [] : (int * int * int) list
```

5)

```
fun dates_in_months (list_of_dates : (int*int*int) list, monthes : int list) =
    if null monthes
        then []
    else dates_in_month(list_of_dates, (hd monthes)) @
dates_in_months(list_of_dates, (tl monthes));

dates_in_months([],[4,5]);
dates_in_months([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],[]);
dates_in_months([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],[4,5]);
dates_in_months([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3),
(1995,4,3)],[4,5,10]);
```

Приклад виконання

```
val dates_in_months = fn :
  (int * int * int) list * int list -> (int * int * int) list
val it = [] : (int * int * int) list
val it = [] : (int * int * int) list
val it = [(1995,4,1),(1995,4,2),(1995,4,3),(1995,5,3)] :
  (int * int * int) list
val it = [(1995,4,1),(1995,4,2),(1995,4,3),(1995,5,3)] :
  (int * int * int) list
```

6)

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

get_nth([],4);
get_nth(["1","2","3","4","5"],4);
```

Приклад виконання

```
val get_nth = fn : string list * int -> string
val it = "" : string
val it = "4" : string
```

7)

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

date_to_string((2003,7,25));
```

Приклад виконання

```
[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 it = "July 25, 2003" : string
```

8)

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

number_before_reaching_sum(4,[1,2,3,4]);
number_before_reaching_sum(4,[]);
```

Приклад виконання

```
val number_before_reaching_sum = fn : int * int list -> int
val it = 2 : int
val it = 0 : int
```

9)

```
fun what_month (day : int) =
    let
        val monthes = [31,28,31,30,31,30,31,31,30,31,30,31]
    in
        number_before_reaching_sum(day, monthes)+1
    end;

what_month(32);
```

Приклад виконання

```
val what_month = fn : int -> int
val it = 2 : int
```

10)

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

month_range(30,33);
```

Приклад виконання

```
val month_range = fn : int * int -> int list
val it = [1,1,2,2] : int list
```

11)

```
fun oldest(dates : (int*int*int) list) =
    if null dates
        then "NONE"
    else
    let
        fun greatest(dates : (int*int*int) list) =
            if (null (tl dates))
                then (hd dates)
            else if (is_older((hd dates),greatest(tl dates)))
                then greatest(tl dates)
                else (hd dates)
    in
        "SOME " ^ date_to_string(greatest(dates))
    end;

oldest([(1995,2,3), (1995,3,3), (1995,4,1), (1995,4,2), (1995,5,3), (1996,4,3)]);
oldest([(1995,2,3)]);
oldest([]);
```

Приклад виконання

```
val oldest = fn : (int * int * int) list -> string
val it = "SOME April 3, 1996" : string
val it = "SOME February 3, 1995" : string
val it = "NONE" : string
```

Висновок: під час виконання даної лабораторної роботи, я ознайомився з сучасним варіантом функціональної мови LISP – SML та написав 11 функцій, відповідно до завдання, на цій мові програмування.