НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №2

з курсу

“Мультипарадигменне програмування”

студента 2 курсу
групи ІП-01
Коваленка Микити Артемовича

Викладач:
ас. Очеретяний О. К.

Київ – 2022

1. **Завдання лабораторної роботи**

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. Виконання роботи

```
(*1*)
fun is_older (date1 : int * int * int, date2: int * int * int) =
  if
    #1 date1 < #1 date2
  orelse
    #1 date1 = #1 date2 andalso #2 date1 < #2 date2
  orelse
    #1 date1 = #1 date2 andalso #2 date1 = #2 date2 andalso #3 date1
< #3 date2
  then true
  else false;

(*2*)
fun number_in_month ([], month) = 0
  | number_in_month (((x : int * int * int)::xs), month) =
    (if #2 x = month then 1 else 0) + number_in_month (xs, month);

(*3*)
fun number_in_months (dates, []) = 0
  | number_in_months (dates, x::xs) = number_in_month (dates, x) +
number_in_months (dates, xs);

(*4*)
fun dates_in_month ([], month) = []
  | dates_in_month ((x : int * int * int)::xs, month) =
    (if #2 x = month then [x] else []) @ dates_in_month (xs, month);

(*5*)
fun dates_in_months (dates, []) = []
  | dates_in_months (dates, (x : int)::xs) = dates_in_month (dates,
x) @ dates_in_months (dates, xs);

(*6*)
fun get_nth ([], _) = ""
```

```
  | get_nth ((x : string)::xs, n) = if n = 1 then x else get_nth (xs,
n - 1);

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

(*7*)
fun date_to_string (date : int * int * int) =
  get_nth (month_names, #2 date) ^ " " ^ Int.toString (#3 date) ^ ",
" ^ Int.toString (#1 date);

(*8*)
fun number_before_reaching_sum (_, []) = 0
  | number_before_reaching_sum (sum, (x::xs)) =
  if x < sum then 1 + number_before_reaching_sum (sum - x, xs) else
0;

(*9*)
fun what_month (day) =
  if day > 0 andalso day < 366 then
    number_before_reaching_sum (day, [31, 28, 31, 30, 31, 30, 31, 31,
30, 31, 30, 31]) + 1   else ~1;

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

(*11*)
fun oldest_date [] = NONE
  | oldest_date [x] = SOME x
  | oldest_date (x::y::xs) = if is_older (x, y) then oldest_date
(x::xs) else oldest_date(y :: xs);
```

3. Тести

```
use "functions.sml";

fun assert (expected, actual) =
    if expected = actual then
        true
    else
        raise Fail "assert failure.";

val dates = [
```

```
    (1986, 5, 26),
    (2019, 6, 12),
    (1770, 12, 5),
    (1950, 1, 8),
    (2007, 3, 19),
    (1822, 5, 2),
    (1999, 12, 31),
    (2000, 1, 1),
    (1492, 8, 24),
    (2004, 7, 30)
];

(*1*)
assert (false, is_older ((2022, 2, 3), (2022, 2, 3)));
assert (true, is_older ((1990, 8, 3), (1990, 9, 12)));
assert (false, is_older ((1789, 12, 4), (1788, 9, 15)));

(*2*)
assert (2, number_in_month (dates, 1));
assert (1, number_in_month (dates, 7));
assert (0, number_in_month (dates, 9));

(*3*)
assert (3, number_in_months (dates, [1, 2, 3]));
assert (2, number_in_months (dates, [10, 11, 12]));
assert (10, number_in_months (dates, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12]));

(*4*)
assert ([], dates_in_month (dates, 11));
assert ([(2007, 3, 19)], dates_in_month (dates, 3));
assert ([(1950, 1, 8), (2000, 1, 1)], dates_in_month (dates, 1));

(*5*)
assert ([], dates_in_months (dates, [9, 10, 11]));
assert ([(1950, 1, 8), (2000, 1, 1), (1986, 5, 26), (1822, 5, 2)],
dates_in_months (dates, [1, 4, 5]));
assert ([(2004, 7, 30)], dates_in_months (dates, [7]));

(*6*)
assert ("April", get_nth (month_names, 4));
assert ("June", get_nth (month_names, 6));
assert ("December", get_nth (month_names, 12));

(*7*)
assert ("December 5, 2020", date_to_string ((2020, 12, 5)));
assert ("July 12, 1872", date_to_string ((1872, 7, 12)));
assert ("October 8, 1234", date_to_string ((1234, 10, 8)));

(*8*)
assert (5, number_before_reaching_sum (11, [3, 2, 5, 0, 0, 100, 23,
1]));
assert (2, number_before_reaching_sum (30, [16, 12, 9, 12, 1]));
assert (0, number_before_reaching_sum (100, [160, 12, 9, 12, 1]));

(*9*)
assert (1, what_month (31));
assert (5, what_month (140));
```

```
assert (11, what_month (334));

(*10*)
assert ([1, 1, 1, 1, 1], month_range (1, 5));
assert ([], month_range (101, 100));
assert ([11, 11, 12, 12], month_range (333, 336));

(*11*)
assert (SOME (1492, 8, 24), oldest_date (dates));
assert (NONE, oldest_date ([]));
```