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

Кафедра інформатики та програмної інженерії

Мультипарадигменне програмування

ЗВІТ

до лабораторної роботи №4

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

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

Київ 2022

Завдання:

1. Напишіть функцію only_capitals яка приймає на вхід string list та повертає string list що має тільки рядки що починаються з Великої літери. Вважайте, що всі рядки мають щонайменше один символ. Використайте List.filter, Char.isUpper, та String.sub щоб створити рішення в 1-2 рядки.

2. Напишіть функцію longest_string1 що приймає string list та повертає найдовший string в списку. Якщо список пустий, поверніть "". У випадку наявності декількох однакових кандидатів, поверніть рядок, що найближче до початку списку. Використайте foldl, String.size, та ніякої рекурсії (окрім як використання foldl що є рекурсивним).

3. Напишіть функцію longest_string2 яка точно така сама як longest_string1 окрім як у випадку однакових кандидатів вона повертає найближчого до кінця кандидата. Ваше рішення має бути майже копією longest_string1. Так само використайте foldl та String.size.

4. Напишіть функції longest_string_helper, longest_string3, та longest_string4 такі що:
• longest_string3 має таку саму поведінку як longest_string1 та longest_string4 має таку саму поведінку як longest_string2.
• longest_string_helper має тип (int * int -> bool) -> string list -> string (зверніть увагу на curry). Ця функція буде схожа на longest_string1 та longest_string2 але вона є більш загальною так як приймає функцію як аргумент.
• Якщо longest_string_helper отримує на вхід функцію яка має поведінку як > (тобто повертає true тоді коли перший аргумент строго більше другого), тоді функція має таку саме поведінку як longest_string1.
• longest_string3 та longest_string4 є визначеними через val-прив'язки і часткове використання longest_string_helper.

5. Напишіть функцію longest_capitalized що приймає на вхід string list та повертає найдовший рядок в списку яка починається з Великої літери , або "" якщо таких рядків немає. Вважайте, що всі рядки мають щонайменше один символ. Використовуйте val-прив'язки та ML бібліотечний o оператор для композиції функцій. Вирішіть проблему з однаковими результатами за прикладом завдання 2.

6. Напишіть функцію rev_string, що приймає на вхід string та повертає string що має ті самі символи в зворотньому порядку. Використайте ML o оператор, бібліотечну функцію rev для перевертання списків, та дві бібліотечні функції з String модулю. (Перегляньте документацію, щоб знайти найкращі підходящі)

Наступні дві проблеми передбачають написання функцій над списками які будуть використані в більш пізніх задачах.

7. Напишіть функцію first_answer типу (’a -> ’b option) -> ’a list -> ’b (зауважте 2 аргументи curry). Перший аргумент має бути застосований до елементів другого аргументу до того моменту, як він поверне SOME v для деякого v і тоді v є результатом виклику first_answer. Якщо перший аргумент повертає NONE для всіх елементів списку, тоді має повернути виключення NoAnswer. Підказка: Приклад розв'язку має 5 рядків і не робить нічого складного.

8. Напишіть функцію all_answers типу (’a -> ’b list option) -> ’a list -> ’b list option (зауважте 2 аргументи curry). Перший аргумент має бути застосований до елементів другого аргументу. Якщо результатом є NONE для будь якого з елементів, то результатом all_answers є NONE. Інакше виклики першого аргументу мають повернути SOME lst1, SOME lst2, ... SOME lstn та результатом all_answers буде SOME lst де lst є lst1, lst2, ..., lstn що складаються разом(порядок не важливий).

Підказки: Приклад розв'язку має 8 рядків. Він використовує допоміжні функції з акумулятором та @. Зауважте all_answers f [] має отримати тип SOME [].
Задачі що залишилися використовують наступні визначення типів, що були створені за образом вбудованої реалізації ML порівняння з шаблоном:
datatype pattern = Wildcard | Variable of string | UnitP | ConstP of int | TupleP of pattern list | ConstructorP of string * pattern
datatype valu = Const of int | Unit | Tuple of valu list | Constructor of string * valu

Дано valu v та pattern p, або p співпадає з v або ні. Якщо так,співпадіння створює список string * valu пар; порядок в списку не має значення. Правила порівняння мають бути наступними:
• Wildcard співпадає з усім і створює пустий список прив’язок.
• Variable s співпадає з будь яким значенням v та створює одно елементний список що містить (s,v).
• UnitP співпадає тільки з Unit та створює пустий список прив’язок.
• ConstP 17 співпадає тільки з Const 17 та створює пустий список прив’язок (так само для інших цілих чисел).
• TupleP ps співпадає з значенням форми Tuple vs якщо ps та vs мають однакову довжину і для всіх i, iий елемент ps співпадає з iим елементом vs. Список прив’язок що створюється в результаті є усіма списками вкладених порівнянь з шаблоном що об’єднані в один список.
• ConstructorP(s1,p) співпадає з Constructor(s2,v) якщо s1 та s2 є однаковою строкою (ви можете порівняти їх з =) та p співпадає з v. Список прив’язок створюється із вкладених порівнянь із шаблоном. Ми називаємо рядки s1 та s2 іменами конструкторів.
• Все інше не має значення.

9. (Ця задача використовує pattern тип даних але не зовсім про порівняння із шаблоном.) Функція g надана в **файлі**.
(1) Використайте g для визначення функції count_wildcards, що приймає на вхід pattern та повертає скільки Wildcard pattern-ів він містить.
(2) Використайте g для визначення функції count_wild_and_variable_lengths що приймає на вхід pattern та повертає кількість Wildcard pattern-ів які він містить плюс

суму довжин рядків всіх змінних що містяться у змінній patterns. (Використайте String.size. Нам важливі тільки імена змінних; імена конструкторів не важливі.)
(3) Використайте g для визначення функції count_some_var що приймає на вхід строку та pattern (як пару) та повертає кількість входжень строки як змінної в pattern. Нам важливі тільки імена змінних; імена конструкторів не важливі.

10. Напишіть функцію check_pat що приймає на вхід pattern та повертає true тоді і тільки тоді коли всі змінні що з'являються в pattern відрізняються один від одного (наприклад, використовують різні рядки). Імена конструкторів не важливі. Підказки: Приклад розв'язку має 2 допоміжні функції. Перша приймає pattern та повертає список всіх рядків які він використовує для змінних. Використовуючи foldl з функцією яка використовує append може бути корисним. Друга функція приймає на вхід список рядків і вирішує чи він має повтори. List.exists може бути корисним. Приклад розв'язку має 15 рядків. Підказка: foldl та List.exists не обов'язкові, але можуть допомогти.

11. Напишіть функцію first_match що приймає на вхід value та список шаблонів та повертає (string * valu) list option, тобто NONE якщо ніякий паттерн зі списку не підходить або SOME lst де lst це список прив'язок для першого паттерну в списку який підійшов. Використайте first_answer та handle-вираз. Підказка: Приклад розв'язку має 3 рядки.

1. Опис програмного коду

task.sml

```
use "hw03.sml";

    (*1*)
fun only_capitals(list_str: string list) =
        List.filter(fn str => Char.isUpper(String.sub(str, 0))) list_str;



    (*2*)
fun longest_string1(list_str: string list) =
        List.foldl(
                    fn(str1, str2) => if (String.size(str2) >= String.size(str1))
                                      then str2
                                      else str1
                ) "" list_str;



    (*3*)
fun longest_string2(list_str: string list) =
        List.foldl(
                    fn(str1, str2) => if (String.size(str2) > String.size(str1))
                                      then str2
                                      else str1
                ) "" list_str;



    (*4*)
fun longest_string_helper f =
        List.foldl(
                    fn(str1, str2) => if (f(String.size(str1),
String.size(str2)))
                                                          then str2
                                                          else str1
                ) "";
val longest_string3 = longest_string_helper( fn(str1, str2) => str2 >= str1 );
val longest_string4 = longest_string_helper( fn(str1, str2)=> str2 > str1 );



    (*5*)
val longest_capitalized  =  longest_string1 o only_capitals;
```

```
    (*6*)
val rev_string = String.implode o List.rev o String.explode;



    (*7*)
fun first_answer f list =
    case list of
        [] => raise NoAnswer
        | hd :: tl => case f(hd) of
                        SOME v => v
                        | NONE => first_answer f tl;



    (*8*)
fun all_answers f list =
    let
        fun fun_help list acc =
            case list of
                [] => SOME acc
                | hd::tl => case f(hd) of
                                NONE => NONE
                                | SOME v => fun_help tl (acc @ v)
    in
        fun_help list []
    end;



    (*9*)

    (*9a*)
fun count_wildcards(p: pattern) =
    g (fn v => 1) (fn v => 0) p


    (*9b*)
fun count_wild_and_variable_lengths(p: pattern) =
     g (fn a => 1) (String.size) p;


    (*9c*)
fun count_some_var(str: string, p: pattern) =
    g (fn a => 0) (fn b => if b = str then 1 else 0) p;



    (*10*)
```

```
fun check_pat (p: pattern) =
    let
        fun get_list_variables(p: pattern) =
            case p of
                Variable x => [x]
                | TupleP tup => List.foldl(fn (p,ps) => ps @
get_list_variables(p)) [] tup
                | ConstructorP(_,p) => get_list_variables(p)
                | _  => []

        fun check_repeats(list: string list) =
            case list of
                [] => true
                | hd::tl => if List.exists(fn a => a = hd) tl
                                then false
                                else check_repeats(tl)
    in
        check_repeats(get_list_variables(p))
    end;



      (*11*)
fun match (value, patern) =
     case (value, patern) of
        (_, Wildcard) => SOME []
        | (_, Variable s) => SOME [(s, value)]
        | (Unit, UnitP) => SOME []
        | (Const v1, ConstP p1) => if v1 = p1
                                   then SOME []
                                   else NONE
        | (Constructor(str, v1), ConstructorP(strP, p1)) => if str = strP
                                                            then match(v1,
p1)
                                                            else NONE
        | (Tuple tupV, TupleP tupP) => if List.length tupV = List.length tupP
                                       then case all_answers
match(ListPair.zip(tupV, tupP)) of
                                            SOME v1 => SOME v1
                                            |_ => NONE
                                       else NONE
        | _ => NONE;
```

hw03.sml (даний в умові задачі файл)

```
exception NoAnswer

datatype pattern = Wildcard
         | Variable of string
```

```
         | UnitP
         | ConstP of int
         | TupleP of pattern list
         | ConstructorP of string * pattern

datatype valu = Const of int
          | Unit
          | Tuple of valu list
          | Constructor of string * valu

fun g f1 f2 p =
    let
    val r = g f1 f2
    in
    case p of
        Wildcard          => f1 ()
      | Variable x        => f2 x
      | TupleP ps         => List.foldl (fn (p,i) => (r p) + i) 0 ps
      | ConstructorP(_,p) => r p
      | _                 => 0
    end
```

3. Тести програми

test.sml

```
3   val t1_1 = only_capitals(["gfggf", "Gsgsg", "Hdhhd"]);
4   val t1_2 = only_capitals([]);
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t1_1 = ["Gsgsg","Hdhhd"] : string list

val t1_2 = [] : string list
```

```
6   val t2_1 = longest_string1(["adc", "ab", "abvf", "addc"]);
7   val t2_2 = longest_string1([]);
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t2_1 = "abvf" : string

val t2_2 = "" : string
```

```
9   val t3_1 = longest_string2(["adc", "ab", "abvf", "addc"]);
10  val t3_2 = longest_string2([]);
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t3_1 = "addc" : string

val t3_2 = "" : string
```

```
12  val t4_1 = longest_string3(["adc", "ab", "abvf", "addc"]);
13  val t4_2 = longest_string4(["adc", "ab", "abvf", "addc"]);
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t4_1 = "abvf" : string

val t4_2 = "addc" : string
```

```
15    val t5_1 = longest_capitalized(["adc", "ab", "abvf", "addc"]);
16    val t5_2 = longest_capitalized(["adc", "Ab", "abvf", "Addc"]);
17
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t5_1 = "" : string

val t5_2 = "Addc" : string
```

```
18    val t6_1 = rev_string("Str");
19    val t6_2 = rev_string("");
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t6_1 = "rtS" : string

val t6_2 = "" : string
```

```
21    val t7_1 = first_answer(fn a => if a > 3 then SOME a else NONE) [2, 0, 3, 6, 5];
22    val t7_2 = first_answer(fn a => if a = 0 then SOME a else NONE) [2, 0, 3, 6, 5];
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t7_1 = 6 : int

val t7_2 = 0 : int
```

```
24    val t8_1 = all_answers(fn a => if a = 2 then NONE else SOME[a]) [1, 2, 4];
25    val t8_2 = all_answers(fn a => if a = 0 then NONE else SOME[a]) [1, 2, 4];
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t8_1 = NONE : int list option

val t8_2 = SOME [1,2,4] : int list option
```

```
27    val t9a_1 = count_wildcards(ConstP 5);
28    val t9a_2 = count_wildcards(TupleP [Wildcard, ConstP 5, UnitP, Wildcard]);
29    val t9a_3 = count_wildcards(TupleP [Wildcard, Wildcard, ConstructorP("Str", TupleP [Wildcard, Wildcard, UnitP]) ] );
30
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t9a_1 = 0 : int

val t9a_2 = 2 : int
```

```
31  val t9b_1 = count_wild_and_variable_lengths(Variable "");
32  val t9b_2 = count_wild_and_variable_lengths(Variable "Str");
33  val t9b_3 = count_wild_and_variable_lengths(TupleP [Wildcard, ConstructorP("Str", TupleP [Wildcard]) ] );
34  val t9b_4 = count_wild_and_variable_lengths(TupleP [Wildcard, ConstructorP("Str", TupleP [Wildcard, Variable "DD"]) ] );
35
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t9b_1 = 0 : int

val t9b_2 = 3 : int

val t9b_3 = 2 : int

val t9b_4 = 4 : int
```

```
36  val t9c_1 = count_some_var("AB", (TupleP ([Variable("ABC")])));
37  val t9c_2 = count_some_var("ABC", (TupleP ([Variable("ABC")])));
38  val t9c_3 = count_some_var("ABC", (TupleP ([Variable("ABC"), TupleP [Wildcard, Variable "ABC"]])));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t9c_1 = 0 : int

val t9c_2 = 1 : int

val t9c_3 = 2 : int
```

```
36  val t10_1 = check_pat(TupleP [Variable "abc", Wildcard, ConstructorP("Str", TupleP [Variable "abc", UnitP]) ] );
37  val t10_2 = check_pat(TupleP [Variable "abc", Wildcard, ConstructorP("Str", TupleP[UnitP]) ] );
38
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t10_1 = false : bool

val t10_2 = true : bool
```

```
39   val t11_1 = match(Unit, Wildcard);
40   val t11_2 = match(Const 1, ConstP 1);
41   val t11_3 = match(Const 1, ConstP 2);
42   val t11_4 = match(Unit, UnitP);
43   val t11_5 = match(Constructor("Str", Unit), ConstructorP("Str", UnitP));
44   val t11_6 = match(Constructor("Str", Unit), ConstructorP("Str2", UnitP));
45   val t11_7 = match(Constructor("Str", Unit), ConstructorP("Str2", Wildcard));
46   val t11_8 = match(Tuple[Unit, Const 1], TupleP[UnitP, ConstP 1]);
47   val t11_9 = match(Tuple[Unit, Const 0], TupleP[UnitP, ConstP 1]);
48   val t11_10 = match(Tuple[Unit, Const 0], Variable "ddd");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val t11_1 = SOME [] : (string * valu) list option

val t11_2 = SOME [] : (string * valu) list option

val t11_3 = NONE : (string * valu) list option

val t11_4 = SOME [] : (string * valu) list option

val t11_5 = SOME [] : (string * valu) list option

val t11_6 = NONE : (string * valu) list option

val t11_7 = NONE : (string * valu) list option

val t11_8 = SOME [] : (string * valu) list option

val t11_9 = NONE : (string * valu) list option

val t11_10 = SOME [("ddd",Tuple [Unit,Const 0])] : (string * valu) list option
```