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 лабораторних робіт

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | ІП-01 Ніколаєв Іван Романович |  |  |
|  |  | (№ 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](https://drive.google.com/file/d/1PdENR1ivx_ng6dIakZEG9456xWn-lkpQ/view?usp=sharing).

(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.

**Програмний код**

**(task.sml)**

val only\_capitals = List.filter (fn str => (Char.isUpper o String.sub) (str, 0))

val longest\_string1 =

    List.foldl (fn (str, acc) =>

           if (String.size str) > (String.size acc)

           then str

           else acc) ""

val longest\_string2 =

    List.foldl (fn (str, acc) =>

           if (String.size str) >= (String.size acc)

           then str

           else acc) ""

fun longest\_string\_helper f =

    List.foldl (fn (str, acc) =>

           if f (String.size str, String.size acc)

           then str

           else acc) ""

val longest\_string3 = longest\_string\_helper (fn (a, b) => a > b)

val longest\_string4 = longest\_string\_helper (fn (a, b) => a >= b)

val longest\_capitalized = longest\_string1 o only\_capitals

val rev\_string = String.implode o rev o String.explode

exception NoAnswer

fun first\_answer f lst =

    case lst of

    [] => raise NoAnswer

      | x :: xs => case f x of

               SOME v => v

             | NONE => first\_answer f xs

fun all\_answers f lst =

    let

    fun all\_answers\_helper remaining acc =

        case (remaining, acc) of

        ([], *\_*) => acc

          | (x :: xs, SOME v) => (case f x of

                      NONE => NONE

                    | SOME xv => all\_answers\_helper xs (SOME (xv @ v)))

          | *\_* => NONE

    in

    all\_answers\_helper lst (SOME [])

    end

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

val count\_wildcards = g (fn *\_* => 1) (fn *\_* => 0)

val count\_wild\_and\_variable\_lengths = g (fn *\_* => 1) String.size

fun count\_some\_var (str, p) = g (fn *\_* => 0) (fn x =>

                        if String.isSubstring str x

                        then 1

                        else 0) p

fun check\_pat p =

    let

    fun filterString pat acc = case pat of

                       Variable x => x :: acc

                     | ConstructorP (*\_*, p) => filterString p acc

                     | TupleP ps =>

                       List.foldl

                       (fn (p, acc) => (filterString p []) @ acc) [] ps

                     | *\_* => []

    in

    let

        val strList = filterString p []

        fun checkDuplicate remList =

        case remList of

            [] => true

          | x :: xs => if List.exists (fn item => item = x) xs

                   then false

                   else checkDuplicate xs

    in

        checkDuplicate strList

    end

    end

fun match (v, p) =

    case p of

    Wildcard => SOME []

      | UnitP => (case v of Unit => SOME []

              | *\_* => NONE)

      | Variable str => SOME [(str, v)]

      | ConstP i => (case v of Const j => if i = j then SOME [] else NONE

                 | *\_* => NONE)

      | TupleP plst => (case v of

                Tuple vlst => if List.length plst = List.length vlst

                      then all\_answers match (ListPair.zip (vlst, plst))

                      else NONE

              | *\_* => NONE)

      | ConstructorP (str, pt) => (case v of

                       Constructor (vstr, vval) => if str = vstr

                                   then match (vval, pt)

                                   else NONE

                     | *\_* => NONE)

fun first\_match v plst =

    SOME (first\_answer (fn p => match (v, p)) plst)

    handle NoAnswer => NONE

**Тести**

**(test.sml)**

use "task.sml"

;

fun test(function\_name : string, true\_result, fact\_result) =

    if true\_result = fact\_result

    then (function\_name, "Ok")

    else (function\_name, "Failed");

;

test("only\_capitals", ["Bx"], only\_capitals(["abc", "aXa", "Bx"]));

test("only\_capitals", [], only\_capitals(["abc", "aXa", "kx"]));

test("longest\_string1", "", longest\_string1([]));

test("longest\_string1", "abcde", longest\_string1(["abcde", "1234", "kx"]));

test("longest\_string1", "abcde", longest\_string1(["abcde", "12345", "kx"]));

test("longest\_string2", "", longest\_string2([]));

test("longest\_string2", "abcde", longest\_string2(["abcde", "1234", "kx"]));

test("longest\_string2", "12345", longest\_string2(["abcde", "12345", "kx"]));

test("longest\_string3", "", longest\_string3([]));

test("longest\_string3", "abcde", longest\_string3(["abcde", "1234", "kx"]));

test("longest\_string3", "abcde", longest\_string3(["abcde", "12345", "kx"]));

test("longest\_string4", "", longest\_string4([]));

test("longest\_string4", "abcde", longest\_string4(["abcde", "1234", "kx"]));

test("longest\_string4", "12345", longest\_string4(["abcde", "12345", "kx"]));

test("longest\_capitalized", "", longest\_capitalized([]));

test("longest\_capitalized", "Abcde", longest\_capitalized(["Abcde", "1234", "kx"]));

test("longest\_capitalized", "", longest\_capitalized(["abcde", "12345", "kx"]));

test("rev\_string", "", rev\_string(""));

test("rev\_string", "edcbA", rev\_string("Abcde"));

test("first\_answer", 5, first\_answer (fn(x) => if x = 5 then SOME 5 else NONE)  [1,2,3,4,5]);

test("first\_answer", 0, first\_answer (fn(x) => if x = 6 then SOME 6 else NONE)  [1,2,3,4,5]);

test("all\_answers", SOME [], all\_answers (fn(x) => if x = 5 then SOME [5] else NONE)  []);

test("all\_answers", NONE, all\_answers (fn(x) => if x = 5 then SOME [5] else NONE)  [1,2,3,4,5]);

test("all\_answers", SOME [5], all\_answers (fn(x) => if x = 5 then SOME [5] else NONE)  [5]);

test("all\_answers", SOME [5,5], all\_answers (fn(x) => if x = 5 then SOME [5] else NONE)  [5,5]);

test("count\_wildcards", 1 ,count\_wildcards Wildcard);

test("count\_wildcards", 1,count\_wildcards (TupleP ([Wildcard])));

test("count\_wildcards", 2,count\_wildcards (TupleP ([Wildcard,Wildcard])));

test("count\_wildcards", 1,count\_wildcards (TupleP ([Wildcard,ConstP(1)])));

test("count\_wildcards", 3,count\_wildcards (TupleP ([Wildcard,Wildcard,Wildcard,ConstP(1)])));

test("count\_wildcards", 0,count\_wildcards (TupleP ([ConstP(1),ConstP(1)])));

test("count\_wild\_and\_variable\_lengths", 3,count\_wild\_and\_variable\_lengths (TupleP ([Wildcard,Wildcard,Wildcard])));

test("count\_wild\_and\_variable\_lengths", 4,count\_wild\_and\_variable\_lengths (TupleP ([Wildcard,Wildcard,Variable("ab")])));

test("count\_wild\_and\_variable\_lengths", 2,count\_wild\_and\_variable\_lengths (TupleP ([Variable("ab")])));

test("count\_wild\_and\_variable\_lengths", 9,count\_wild\_and\_variable\_lengths (TupleP ([Variable("ab"),Variable"abcde",Wildcard,Wildcard])));

test("count\_some\_var", 1,count\_some\_var("ab", (TupleP ([Variable("ab")]))));

test("count\_some\_var", 2,count\_some\_var("ab", (TupleP ([Variable("ab"),Variable("bc"),Variable("ab")]))));

test("count\_some\_var", 0,count\_some\_var("ab2", (TupleP ([Variable("ab"),Variable("bc"),Variable("ab")]))));

test("count\_some\_var", 0,count\_some\_var("wild",ConstructorP ("wild",(Wildcard))));

test("count\_some\_var", 2,count\_some\_var ("x",TupleP[TupleP[TupleP[Variable "x",ConstructorP("wild",Wildcard)],Wildcard],Variable "x"]));

test("check\_pat", true, check\_pat((TupleP ([Variable("")]))));

test("check\_pat", true, check\_pat((TupleP ([Variable("ab")]))));

test("check\_pat", false, check\_pat((TupleP ([Variable("ab"),Variable("ab"),Variable("bc")]))));

test("check\_pat", true, check\_pat( (TupleP ([Variable("ab1"),Variable("ab2"),Variable("ab3")]))));

test("match", NONE, match( Const 1,TupleP ([Variable("")])));

test("match", SOME [], match( Const 1,ConstP 1));

test("match", SOME[], match( Tuple[ (Const 1), Unit,Constructor("asd", Const 1), Constructor("dsa",Const 2)],

                            TupleP ([ConstP 1,UnitP,ConstructorP ("asd", ConstP 1),ConstructorP("dsa",ConstP 2) ])));

test("match", SOME [("dsa3",Constructor("dsa",Const 2)),("dsa2",Constructor("asd", Const 1)),

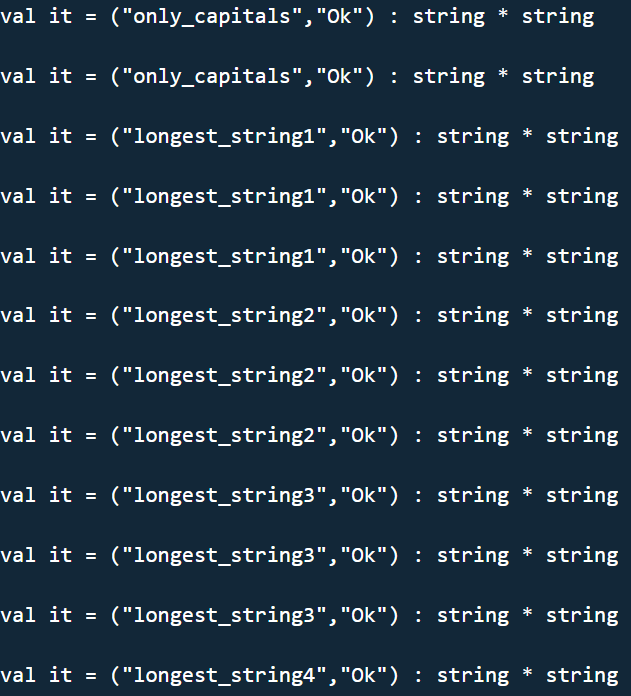
                                           ("dsa1",Const 1)], match( Tuple[ (Const 1), Unit,Constructor("asd", Const 1), Constructor("dsa",Const 2)],

                            TupleP ([Variable("dsa1"), UnitP , Variable("dsa2") ,Variable("dsa3")])));

test("first\_match", NONE, first\_match (Constructor("dsa",Const 1)) ([UnitP,ConstP 1]));

test("first\_match", SOME [("abc",Const 1)], first\_match (Const 1) ([UnitP,Variable("abc")]));

test("first\_match", SOME [("abc",Const 1)], first\_match (Const 1) ([Variable("abc"),Variable("abc1")]));

****

