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

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи №4

Виконав студент *ІП-01 Смислов Даніл* _____
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

1. Завдання лабораторної роботи

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.

2. Програмний код

hw03.sml

```
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


(**** you can put all your code here ****)
```

Lab4.sml

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

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

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

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

val longest_string3 = longest_string_helper(fn(str1,str2)=>if(str1>str2) then true else
false);
val longest_string4 = longest_string_helper(fn(str1,str2)=>if(str1>=str2) then true else
false);

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

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

(*7*)
exception NoAnswer;
fun first_answer f lst =
case lst of
    [] => raise NoAnswer
```

```
    |hd::tl => case f(hd) of
        NONE => first_answer f tl
        |SOME x => x

(*8*)
fun all_answers f lst=
    let fun internal(acc,lst) =
        case lst of
            hd::tl =>
                (case f(hd) of
                    SOME x => internal (acc @ x, tl)
                    |NONE => NONE)
            |[] => SOME(acc)
    in
    internal([],lst)
    end;

(*9*)
use "hw03.sml";

(*9a*)
fun count_wildcards(pat:pattern)=
    g (fn _ => 1) (fn _ => 0) pat;

(*9b*)
fun count_wild_and_variable_lengths(pat:pattern)=
    g (fn _ => 1) (fn x => String.size x) pat;

(*9c*)
fun count_some_var(str : string, pat : pattern)=
    g (fn _ => 0) ( fn x => if x = str then 1 else 0) pat;

(*10*)
fun check_pat(pat: pattern)=
    let
        fun getStrings(pat: pattern) =
        case pat of
            Variable x => [x]
            |TupleP tuple=> List.foldl(fn(p,acc)=> getStrings(p) @ acc) [] tuple
            |ConstructorP(_,p) => getStrings(p)
            | _ => []
        fun checkRep(lst: string list) =
            case lst of
                [] => true
                |hd::tl => if (List.exists(fn el => el = hd) tl)
                        then false
                        else checkRep(tl)
        in
            checkRep(getStrings(pat))
        end;
```

3. Приклади виконання програми

Tests.sml

Tests.sml

```
1   use "Lab4.sml";
2   (*1*)
3   val test1 = only_capitals(["Check2","check1","check3"]); (*["Check2"]*)
4   val test2 = only_capitals(["check2","check1","check3"]); (*[]*)
5
6   (*2*)
7   val test3 = longest_string1(["check123","check124","check3"]); (*"check123"*)
8   val test4 = longest_string1([]); (*""*)
9
10  (*3*)
11  val test5 = longest_string2(["check123","check124","check3"]); (*"check124"*)
12
13  (*4*)
14  val test6 = (longest_string3(["check123","check124","check3"])); (*"check123"*)
15  val test7 = (longest_string3([])); (*""*)
16  val test8 = (longest_string4(["check123","check124","check3"])); (*"check124"*)
17  val test9 = (longest_string4([])); (*""*)
18
19  (*5*)
20  val test10 =  longest_capitalized(["Check123","Check124","check3"]); (*"Check123"*)
21  val test11 =  longest_capitalized([]); (*""*)
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
val test1 = ["Check2"] : string list

val test2 = [] : string list

val test3 = "check123" : string

val test4 = "" : string

val test5 = "check124" : string

val test6 = "check123" : string

val test7 = "" : string

val test8 = "check124" : string

val test9 = "" : string

val test10 = "Check123" : string

val test11 = "" : string
```

```
23  (*6*)
24  val test12 = rev_string("check1"); (*"1kcehc"*)
25  val test13 = rev_string("");
26
27  (*7*)
28  val test14 = first_answer (fn el => if Char.isUpper(String.sub(el,0))then SOME el else NONE)["Check123","Check124","check3"]; (*"Check123"*)
29
30  (*8*)
31  val test15 = all_answers(fn el => if Char.isUpper(String.sub(el,0))then SOME [el] else NONE)["Check123","Check124","Check3"]; (*SOME ["Check123","Check124","Check3"]*)
32  val test16 = all_answers(fn el => if Char.isUpper(String.sub(el,0))then SOME [el] else NONE)["Check123","check124","Check3"]; (*NONE*)
33
34
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
val test12 = "1kcehc" : string

val test13 = "" : string

val test14 = "Check123" : string

val test15 = SOME ["Check123","Check124","Check3"] : string list option

val test16 = NONE : string list option
```

```
35  (*9*)
36  val test17 = count_wildcards(TupleP[Wildcard,Wildcard]); (*2*)
37  val test18 = count_wildcards(TupleP[Wildcard,Wildcard, TupleP[Wildcard,Wildcard,Wildcard]]); (*5*)
38  val test19 = count_wild_and_variable_lengths(TupleP[Wildcard,Wildcard,Variable "check"]); (*7*)
39  val test20 = count_wild_and_variable_lengths(Variable ""); (*0*)
40  val test21 = count_some_var("check",TupleP[Wildcard,Wildcard,Variable "check",Variable "check2",Variable "check"]); (*2*)
41  val test22 = count_some_var("check3",TupleP[Wildcard,Wildcard,Variable "check",Variable "check2",Variable "check"]); (*0*)
42
43  (*10*)
44  val test23 = check_pat(TupleP [Variable "check", Wildcard]); (*true*)
45  val test24 = check_pat(TupleP [Variable "check", Wildcard, TupleP[Variable "check"]]); (*false*)
46  val test25 = check_pat(TupleP [Wildcard,Wildcard, TupleP[Wildcard,Wildcard,Wildcard]]); (*true*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
val test17 = 2 : int

val test18 = 5 : int

val test19 = 7 : int

val test20 = 0 : int

val test21 = 2 : int

val test22 = 0 : int

val test23 = true : bool

val test24 = false : bool

val test25 = true : bool
```