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 № 4 з дисципліни

“Мультипарадигменне програмування”

Виконав студент: ІП-01 Хернуф Валід

Перевірив: ас. Очеретяний О. К.

Київ 2022

1. Завдання лабораторної роботи

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 і, іий елемент `ps` співпадає з іим 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

fourthlab.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

(*1 task*)
fun only_capitals(str_lst) =
    List.filter(fn
        "" => false
        | str => Char.isUpper(String.sub(str, 0))
        ) str_lst

(*2 task*)
fun longest_string1(str_lst) =
    List.foldl(fn (cur_str, long_str) =>
        if String.size cur_str > String.size long_str
        then cur_str
        else long_str
        ) "" str_lst

(*3 task*)
fun longest_string2(str_lst) =
    List.foldl(fn (cur_str, long_str) =>
        if String.size cur_str >= String.size long_str
        then cur_str
        else long_str
```

```
        ) "" str_lst

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

val longest_string3 = longest_string_helper (fn (str_1, str_2 ) => str_1 > str_2)

val longest_string4 = longest_string_helper (fn (str_1, str_2 ) => str_1 >=
str_2)

(*5 task*)
val longest_capitalized = longest_string3 o only_capitals

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

(*7 task*)
fun first_answer func ([]) = raise NoAnswer
  | first_answer func (h::t) =
    case func (h) of
    SOME v => v
    | NONE => first_answer func t

(*8 task*)
fun all_answers func lst =
    let fun tail_rec([], a) = SOME(a)
          | tail_rec(h::t, a) =
                case func(h) of
                SOME v => tail_rec(t, a @ v)
                | NONE => NONE
    in tail_rec(lst, []) end

(*9 task*)

(*9.1 task*)
val count_wildcards = g (fn _ => 1) (fn _ => 0)

(*9.2 task*)
val count_wild_and_variable_lengths = g (fn _ => 1) (String.size)

(*9.3 task*)
fun count_some_var(str, pat) = g (fn _ => 0) (fn sp => if sp = str then 1 else 0)
pat

(*10 task*)
fun check_pat(pat) =
```

```
    let fun getTypes(Variable x) = [x]
          | getTypes(ConstructorP(_, pa1)) = getTypes(pa1)
          | getTypes(TupleP ps) = List.foldl(fn (pa2, a) => getTypes(pa2) @ a) []
ps
          | getTypes(_) = []
        fun isDublicates([]) = true
          | isDublicates(h::t) =
                if List.exists(fn s => s = h) t
                then false
                else isDublicates(t)
    in isDublicates(getTypes(pat)) end

(*11 task*)
fun first_match (_, Wildcard) = SOME []
  | first_match (Unit, UnitP) = SOME []
  | first_match (v , Variable s) = SOME [(s, v)]
  | first_match (Const v, ConstP p) =
        if v = p
        then SOME []
        else NONE
  | first_match (Constructor(str, v), ConstructorP(str_p, p)) =
        if str = str_p
        then first_match(v, p)
        else NONE
  | first_match (Tuple v, TupleP p) =
        if List.length v = List.length p
        then case all_answers first_match(ListPair.zip(v, p)) of
                  SOME v1 => SOME v1
                  |_ => NONE
        else NONE
  | first_match (_, _) = NONE
```

3. Скріншоти роботи функцій

Тестування функції 1:

```
fun provided_test1() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "", "errors"]
        val strList3 = []
    in
        (only_capitals(strList1),
        only_capitals(strList2),
        only_capitals(strList3))
    end

val ans_first = provided_test1()
```

Результат тестування функції 1:

```
val ans_first = (["Hello","My","Name","Is","Valid"],["Checks"],[]) :
  string list * string list * string list
```

Тестування функції 2:

```
fun provided_test2() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "errors", "srorre"]
        val strList3 = []
    in
        (longest_string1(strList1),
        longest_string1(strList2),
        longest_string1(strList3))
    end

val ans_second = provided_test2()
```

Результат тестування функції 2:

```
val ans_second = ("robert","Checks","") : string * string * string
```

Тестування функції 3:

```
fun provided_test3() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "errors", "srorre"]
        val strList3 = []
    in
        (longest_string2(strList1),
        longest_string2(strList2),
        longest_string2(strList3))
    end

val ans_third = provided_test3()
```

Результат тестування функції 3:

```
val ans_third = ("robert","srorre","") : string * string * string
```

Тестування функції 4:

```
fun provided_test4() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "errors", "srorre"]
        val strList3 = []
    in
        (longest_string3(strList1),
        longest_string3(strList2),
        longest_string3(strList3))
    end

val ans_fourth = provided_test4()
```

Результат тестування функції 4:

```
val ans_fourth = ("robert","Checks","") : string * string * string
```

Тестування функції 5:

```
fun provided_test5() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "errors", "srorre"]
        val strList3 = []
    in
        (longest_string4(strList1),
        longest_string4(strList2),
        longest_string4(strList3))
    end

val ans_fifth = provided_test5()
```

Результат тестування функції 5:

```
val ans_fifth = ("robert","srorre","") : string * string * string
```

Тестування функції 6:

```
fun provided_test6() =
    let val strList1 = ["Hello", "My", "Name", "Is", "robert", "Valid"]
        val strList2 = ["", "Checks", "for", "errors", "srorre"]
        val strList3 = []
    in
        (longest_capitalized(strList1),
        longest_capitalized(strList2),
        longest_capitalized(strList3))
    end

val ans_sixth = provided_test6()
```

Результат тестування функції 6:

```
val ans_sixth = ("Hello","Checks","") : string * string * string
```

Тестування функції 7:

```
fun provided_test7() =
    let val str1 =  "cat"
        val str2 = "reversable"
        val str3 = ""
    in
        (rev_string(str1),
        rev_string(str2),
        rev_string(str3))
    end

val ans_seventh = provided_test7()
```

Результат тестування функції 7:

```
val ans_seventh = ("tac","elbasrever","") : string * string * string
```

Тестування функції 8:

```
fun provided_test8() =
    let
        val l1 = [5]
        val l2 = [2, 4, 6, 8]
        val l3 = [10, 20, 30]
    in
        ((first_answer (fn a => if a = 5 then SOME a else NONE) l1) handle
(NoAnswer) => 0,
        (first_answer (fn a => if a = 10 then SOME a else NONE) l2) handle
(NoAnswer) => 0,
        (first_answer (fn a => if a = 20 then SOME a else NONE) l3) handle
(NoAnswer) => 0)
    end

val ans_eighth = provided_test8()
```

Результат тестування функції 8:

```
val ans_eighth = (5,0,20) : int * int * int
```

Тестування функції 9:

```
fun provided_test9() =
    let
        val list1 = [3, 3, 3]
        val list2 = [3, 2, 3]
        val list3 = [10, 20, 30]
    in
        (all_answers (fn c => if c = 3 then SOME [c] else NONE) list1,
        all_answers (fn c => if c = 3 then SOME [c] else NONE) list2,
        all_answers (fn c => if c > 5 then SOME [c] else NONE) list3)
    end

val ans_ninth = provided_test9()
```

Результат тестування функції 9:

```
val ans_ninth = (SOME [3,3,3],NONE,SOME [10,20,30]) :
  int list option * int list option * int list option
```

Тестування функції 10:

```
fun provided_test10() =
    let
        val p1: pattern = Wildcard
        val p2: pattern = Variable "hello"
        val p3: pattern = (TupleP ([Wildcard,ConstP(1), Wildcard, Wildcard]))
        val p4: pattern = (TupleP ([ConstP(1),ConstP(1), ConstructorP("mystr",
Wildcard)]))
    in
       (count_wildcards(p1),
        count_wildcards(p2),
        count_wildcards(p3),
        count_wildcards(p4))
    end

val ans_tenth = provided_test10()
```

Результат тестування функції 10:

```
val ans_tenth = (1,0,3,1) : int * int * int * int
```

Тестування функції 11:

```
fun provided_test11() =
    let
        val p1: pattern = Wildcard
        val p2: pattern = Variable "txt"
        val p3: pattern = (TupleP ([Variable "some", Variable "something else",
ConstructorP("str", Wildcard)]))
        val p4: pattern = (TupleP ([Wildcard, Variable "check", Wildcard,
Wildcard, Variable "second check"]))
    in
       (count_wild_and_variable_lengths(p1),
        count_wild_and_variable_lengths(p2),
        count_wild_and_variable_lengths(p3),
        count_wild_and_variable_lengths(p4))
    end

val ans_eleventh = provided_test11()
```

Результат тестування функції 11:

```
val ans_eleventh = (1,3,19,20) : int * int * int * int
```

Тестування функції 12:

```
fun provided_test12() =
    let
        val p1: pattern = Wildcard
        val p2: pattern = Variable "txt"
        val p3: pattern = (TupleP ([Wildcard, Variable "check", Wildcard,
Wildcard]))
        val p4: pattern = (TupleP ([Variable "some", Variable "some",
ConstructorP("str", Wildcard)]))
    in
       (count_some_var("nothing", p1),
        count_some_var("txt", p2),
        count_some_var("some", p3),
        count_some_var("check", p4))
    end

val ans_twelfth = provided_test12()
```

Результат тестування функції 12:

```
val ans_twelfth = (0,1,0,0) : int * int * int * int
```

Тестування функції 13:

```
fun provided_test13() =
    let
        val p1: pattern = Wildcard
        val p2: pattern = Variable "txt"
        val p3: pattern = (TupleP ([Variable "some", Variable "some",
ConstructorP("str", Wildcard)]))
        val p4: pattern = (TupleP ([Wildcard, Variable "second check", Wildcard,
Wildcard, Variable "check"]))
    in
       (check_pat(p1),
        check_pat(p2),
        check_pat(p3),
        check_pat(p4))
    end

val ans_thirteenth = provided_test13()
```

Результат тестування функції 13:

```
val ans_thirteenth = (true,true,false,true) : bool * bool * bool * bool
```

Тестування функції 14:

```
fun provided_test14() =
    (first_match(Unit, UnitP),
    first_match(Unit, Wildcard),
    first_match(Const 1, ConstP 1),
    first_match(Const 1, ConstP 2),
    first_match(Tuple[Unit, Const 0], Variable "txt"),
    first_match(Tuple[Unit, Const 1], TupleP[UnitP, ConstP 1]),
    first_match(Tuple[Unit, Const 0], TupleP[UnitP, ConstP 1]),
    first_match(Constructor("SomeTxt", Unit), ConstructorP("SomeTxt", UnitP)),
    first_match(Constructor("SomeTxt", Unit), ConstructorP("SomeOtherTxt",
UnitP)),
    first_match(Constructor("SomeTxt", Unit), ConstructorP("SomeOtherTxt",
Wildcard)))

val ans_fourteenth = provided_test14()
```

Результат тестування функції 14:

```
val ans_fourteenth =
  (SOME [],SOME [],SOME [],NONE,SOME [("txt",Tuple [Unit,Const 0])],SOME [],
   NONE,SOME [],NONE,NONE) :
  (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
```

Цілий скріншот результатів:

```
val ans_first = (["Hello","My","Name","Is","Valid"],["Checks"],[]) :
  string list * string list * string list
val ans_second = ("robert","Checks","") : string * string * string
val ans_third = ("robert","srorre","") : string * string * string
val ans_fourth = ("robert","Checks","") : string * string * string
val ans_fifth = ("robert","srorre","") : string * string * string
val ans_sixth = ("Hello","Checks","") : string * string * string
val ans_seventh = ("tac","elbasrever","") : string * string * string
val ans_eighth = (5,0,20) : int * int * int
val ans_ninth = (SOME [3,3,3],NONE,SOME [10,20,30]) :
  int list option * int list option * int list option
val ans_tenth = (1,0,3,1) : int * int * int * int
val ans_eleventh = (1,3,19,20) : int * int * int * int
val ans_twelfth = (0,1,0,0) : int * int * int * int
val ans_thirteenth = (true,true,false,true) : bool * bool * bool * bool
val ans_fourteenth =
  (SOME [],SOME [],SOME [],NONE,SOME [("txt",Tuple [Unit,Const 0])],SOME [],
   NONE,SOME [],NONE,NONE) :
  (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
   * (string * valu) list option * (string * valu) list option
-
```