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 Шпилька Владислав Сергійович _____
(№ групи, прізвище, ім'я, по батькові)

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

Київ 2021

Завдання:

Завдання 1:

Це завдання пов'язане з використанням “заміни імені”, щоб придумати альтернативні імена. Наприклад, Фредерік Вільям Сміт також може бути Фредом Вільямом Смітом або Фредді Вільямом Смітом. Тільки частина (d) присвячена цьому, але інші проблеми є корисними.

(a) Напишіть функцію all_except_option, яка приймає string і string list. Поверніть NONE, якщо рядка немає у списку, інакше поверніть SOME lst, де lst ідентичний списку аргументів, за винятком того, що рядка в ньому немає. Ви можете вважати, що рядок є в списку щонайбільше один раз. Використовуйте рядок, наданий вам, для порівняння рядків. Приклад рішення становить близько 8 строк.

(b) Напишіть функцію get_substitutions1, яка приймає string list list (список списків рядків, замін) і string s і повертає string list. Результат містить всі рядки, які є в якомусь із списків замін, які також мають s, але сам s не повинен бути в результаті.
приклад: get_substitutions1([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], «Fred»)
відповідь: ["Fredrick","Freddie","F"]
Припустимо, що кожен список із замінами не має повторів. Результат повторюватиметься, якщо s та інший рядок є в більш ніж одному списку підстановок. приклад:
get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Jeff")
(* відповідь: ["Jeffrey","Geoff","Jeffrey"] *)
Використовуйте підзадачу (a) і додавання до списку ML (@), але ніяких інших допоміжних функцій. Зразок рішення становить близько 6 рядків.

(c) Напишіть функцію get_substitutions2, схожу на get_substitutions1, за винятком того, що вона використовує хвостову рекурсивну локальну допоміжну функцію.
(d) Напишіть функцію `similar_names`, яка приймає `string list list` із підстановками (як у частинах (b) і (c)) і *повне ім'я* типу {first:string,middle:string,last:string} і повертає список повних імен (тип {first:string,middle:string,last:string} list). Результатом є всі *повні імена*, які ви можете створити, замінивши ім’я (і лише ім’я), використовуючи заміни та частини (b) або (c). Відповідь має починатися з оригінальної назви (тоді мати 0 або більше інших імен).
Приклад: `similar_names`([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], {first="Fred", middle="W", last="Smith"})
відповідь:

```
[{first="Fred", last="Smith", middle="W"},
{first="Fredrick", last="Smith", middle="W"},
{first="Freddie", last="Smith", middle="W"},
{first="F", last="Smith", middle="W"}]
```

Не видаляйте дублікати з відповіді. Підказка: використовуйте локальну допоміжну функцію. Зразок рішення становить близько 10 рядків.

Завдання 2:

У цій задачі йдеться про карткову гру-пасьянс, придуману саме для цього питання. Ви напишете програму, яка відстежує хід гри. Ви можете виконати частини (a)–(e), перш ніж зрозуміти гру, якщо хочете. Гра проводиться з *колодою карт* і ціллю. У гравця є *список карт в руці*, спочатку порожній. Гравець робить хід, витягуючи карту з *колоди*, що означає вилучення першої карти зі *списку карт колоди* і додавання її до списку *карт в руці*, або скидання, що означає вибір однієї з *карт в руці* для видалення. Гра закінчується або тоді, коли гравець вирішує більше не робити ходів, або коли сума значень утриманих карт перевищує ціль.

Ціль – закінчити гру з низьким результатом (0 найкращий результат). Підрахунок балів працює наступним чином: Нехай sum — це сума значень карт, що в руці. Якщо sum більша за goal, *попередній рахунок* = 3*(sum − goal), інакше *попередній рахунок* = (goal − sum). Кінцевий рахунок дорівнює *попередньому рахунку*, якщо всі картки, які на руці, не однакового кольору. Якщо всі картки одного кольору, кінцевий рахунок є попереднім рахунком, поділеним на 2 (і округлений, за допомогою цілочисельного ділення; використовуйте оператор div ML)

(a) Напишіть функцію card_color, яка бере карту і повертає її колір (піки і трефи чорні, бубни і чирви червоні). Примітка: достатньо одного case-виразу.

(b) Напишіть функцію card_value, яка бере карту та повертає її значення (нумеровані карти мають свій номер як значення, тузи — 11, все інше — 10). Примітка: достатньо одного case-виразу.

(c) Напишіть функцію remove_card, яка бере список карт *cs*, картку *c* та виняток *e*. Функція повертає список, який містить усі елементи *cs*, крім *c*. Якщо *c* є у списку більше одного разу, видаліть лише перший. Якщо *c* немає у списку, поверніть виняток *e*. Ви можете порівнювати карти з =.

(d) Напишіть функцію all_same_color, яка приймає список карт і повертає true, якщо всі карти в списку мають однаковий колір.

(e) Напишіть функцію sum_cards, яка бере список карт і повертає суму їх значень. Використовуйте локально визначену допоміжну функцію, яка є хвостово-рекурсивною.

(f) Напишіть функцію score, яка отримує на вхід card list (картки, що утримуються) та int (ціль) і обчислює рахунок, як описано вище.

(g) Напишіть функцію officiate, яка «запускає гру». Вона приймає на вхід card list (список карт), move list (що гравець «робить» у кожній точці) та int (ціль) і повертає рахунок у кінці гри після обробки (частину чи всі) переміщення в списку переміщень по порядку. Використовуйте локально визначену рекурсивну допоміжну функцію, яка приймає кілька аргументів, які разом представляють поточний стан гри. Як описано вище:

• Гра починається з того, що утримувані карти є порожнім списком.

• Гра закінчується, якщо більше немає ходів. (Гравець вирішив зупинитися, оскільки move list порожній.)

• Якщо гравець скидає якусь карту c, гра продовжується (тобто виконується рекурсивний виклик), коли утримувані карти не мають c, а список карт залишається незмінним. Якщо c немає в картках, що утримуються, поверніть виняток IllegalMove.

• Якщо гравець бере, але список карт (уже) порожній, гра закінчена. Інакше, якщо розіграш призведе до того, що сума карт, що тримаються, перевищує ціль, гра закінчується (після розіграшу). В іншому випадку гра продовжується з більшою кількістю карт на руці та меншою колодою.

Типове рішення для (g) містить менше 20 рядків.

Реалізація:

```
(* if you use this function to compare two strings (returns true if the same
   string), then you avoid several of the functions in problem 1 having
   polymorphic types that may be confusing *)
fun same_string(s1 : string, s2 : string) =
    s1 = s2

(* a *)
fun all_except_option(str, strlist) =
    let fun help_fun(strlist, acc, isFound) =
        case strlist of
            [] => (acc, isFound)
            |(x::xs) => if (same_string(x, str)) then help_fun(xs, acc, true)
else help_fun(xs, x::acc, isFound)
    in
        let fun rev(lst, acc) =
            case lst of
            [] => acc
            | x::xs => rev(xs, x::acc)
        in
            case help_fun(strlist, [], false) of
            (_, false) => NONE
            | ([], true) => SOME([])
            | (x::xs, true) => SOME(rev(x::xs, []))
        end
    end


(* b *)
fun get_substitutions1(strlists, str) =
    case strlists of
    [] => []
    |(x::xs) =>
        case all_except_option(str, x) of
        SOME (y::ys) =>  (y::ys) @ get_substitutions1(xs, str)
        | _ =>  get_substitutions1(xs, str)


(* c *)
fun get_substitutions2(strlists, str) =
    let fun help_fun(strlists, acc) =
        case strlists of
        [] => acc
        |(x::xs) =>
            case all_except_option(str, x) of
            SOME (y::ys) =>  help_fun(xs, acc @ (y::ys))
            | _ =>  help_fun(xs, acc)
    in
```

```
        help_fun(strlists, [])
    end


(* d *)
fun similar_names(strlists, fullname) =
    let fun help_fun(simnames) =
        case simnames of
        [] => []
        |(l::ls) =>
            case fullname of
           {first=x,middle=y,last=z} => {first=l, middle=y, last=z} ::
help_fun(ls)
    in
        case fullname of
            {first=x,middle=y,last=z} =>
fullname::help_fun(get_substitutions2(strlists, x))
    end


(* you may assume that Num is always used with values 2, 3, ..., 10
   though it will not really come up *)
datatype suit = Clubs | Diamonds | Hearts | Spades
datatype rank = Jack | Queen | King | Ace | Num of int
type card = suit * rank

datatype color = Red | Black
datatype move = Discard of card | Draw

exception IllegalMove

(* put your solutions for problem 2 here *)

(* a *)
fun card_color(suit, rank) =
    case suit of
        Diamonds => Red
        |Hearts => Red
        | _ => Black


(* b *)
fun card_value(suit, rank) =
    case rank of
        Num n => n
        |Ace => 11
        | _ => 10


(* c *)
fun remove_card(cs, c, e) =
```

```
    let fun help_fun(cs, acc, isFound) =
        case cs of
            [] => (acc, isFound)
            |(x::xs) => if x = c then (acc @ xs, true) else help_fun(xs, x::acc,
isFound)
    in
     case help_fun(cs, [], false) of
            (_, false) => raise e
            | ([], true) => []
            | (x::xs, true) => x::xs
    end


(* d *)
fun all_same_color [] = true
    |all_same_color (x::[]) = true
    |all_same_color (x::y::[]) = (card_color(x) = card_color(y))
    |all_same_color (x::y::xs) = ((card_color(x) = card_color(y)) andalso
all_same_color(xs))

(* e *)
fun sum_cards(cs) =
    let fun help_fun(cs, sum) =
    case cs of
        [] => sum
        |(x::xs) => help_fun(xs, sum + card_value(x))
    in
        help_fun(cs, 0)
    end


(* f *)
fun score(cs, goal) =
    let
        val sum = sum_cards(cs)
        val previous_sum = if sum > goal then 3*(sum - goal) else goal - sum
    in
        if (all_same_color(cs)) then
            previous_sum div 2
        else
            previous_sum
    end

(* g *)
fun officiate(cs, ms, goal) =
    let
    fun current_state(cs, ms, players_cards) =
        case ms of
        [] => score(players_cards, goal)
        |move::moves =>
            case move of
```

```
            Discard c => current_state(cs, moves, remove_card(players_cards, c,
IllegalMove))
            |Draw =>
                case cs of
                [] => score(players_cards, goal)
                |x::xs =>
                    if sum_cards(x::players_cards) < goal then
                        current_state(xs, moves, x::players_cards)
                    else
                        score(x::players_cards, goal)
    in
    current_state(cs, ms, [])
    end
```

Тести:

```
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");



(* 1 *)
(* a *)
test("all_except_option", SOME ["1", "3", "4"], all_except_option("2", ["2", "1",
"3", "4"]));
test("all_except_option", SOME [], all_except_option("2", ["2"]));
test("all_except_option", NONE, all_except_option("2", ["1", "3", "4"]));


(* b *)
test("get_substitutions1", ["Fredrick","Freddie","F"],
get_substitutions1([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","
F"]], "Fred"));
test("get_substitutions1", ["Jeffrey","Geoff","Jeffrey"],
get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffr
ey"]], "Jeff"));
test("get_substitutions1", [],
get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"]], "Vlad"));


(* c *)
test("get_substitutions2", ["Fredrick","Freddie","F"],
get_substitutions2([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","
F"]], "Fred"));
test("get_substitutions2", ["Jeffrey","Geoff","Jeffrey"],
get_substitutions2([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffr
ey"]], "Jeff"));
test("get_substitutions2", [],
get_substitutions2([["Fred","Fredrick"],["Jeff","Jeffrey"]], "Vlad"));


(* d *)
test("similar_names",
    [{first="Fred", last="Smith", middle="W"},
        {first="Fredrick", last="Smith", middle="W"},
        {first="Freddie", last="Smith", middle="W"},
        {first="F", last="Smith", middle="W"}],
    similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F
"]], {first="Fred", middle="W", last="Smith"}));
```

```
test("similar_names",
    [{first="Fred", last="Smith", middle="W"}],
    similar_names([["Fred"],["Elizabeth","Betty"]], {first="Fred", middle="W",
last="Smith"}));

test("similar_names",
    [{first="Fred", last="Smith", middle="W"}],
    similar_names([["Elizabeth","Betty"]], {first="Fred", middle="W",
last="Smith"}));



(* 2 *)
val test_card1 = (Hearts, Jack);
val test_card2 = (Clubs, Num 8);
val test_card3 = (Diamonds, Ace);

val test_card_list1 = [test_card1, test_card2, test_card3];
val test_card_list2 = [test_card1, test_card3];
val test_card_list3 = [];


(* a *)
test("card_color", Red, card_color(test_card1));
test("card_color", Black, card_color(test_card2));
test("card_color", Red, card_color(test_card3));

(* b *)
test("card_value", 10, card_value(test_card1));
test("card_value", 8, card_value(test_card2));
test("card_value", 11, card_value(test_card3));


(* c *)
test("remove_card", [test_card2, test_card3], remove_card(test_card_list1,
test_card1, IllegalMove));
test("remove_card", [], remove_card(test_card_list2, test_card2, IllegalMove));
test("remove_card", [], remove_card(test_card_list3, test_card2, IllegalMove));


(* d *)
test("all_same_color", false, all_same_color(test_card_list1));
test("all_same_color", true, all_same_color(test_card_list2));
test("all_same_color", true, all_same_color(test_card_list3));


(* e *)
test("sum_cards", 29, sum_cards(test_card_list1));
test("sum_cards", 21, sum_cards(test_card_list2));
test("sum_cards", 0, sum_cards(test_card_list3));
```

```
(* f *)
test("score", 3, score(test_card_list1, 28));
test("score", 1, score(test_card_list2, 23));
test("score", 2, score(test_card_list1, 31));

(* g *)
test("officiate", 3, officiate(test_card_list1, [Draw, Draw, Draw, Draw], 28));
test("officiate", 5, officiate(test_card_list1, [Draw, Draw, Discard(Clubs, Num
8)], 20));
test("officiate", 9, officiate(test_card_list1, [Draw, Draw], 15));
```

```
[opening task.sml]
task.sml:99.30 Warning: calling polyEqual

val same_string = fn : string * string -> bool
val all_except_option = fn : string * string list -> string list option
val get_substitutions1 = fn : string list list * string -> string list
val get_substitutions2 = fn : string list list * string -> string list
val similar_names = fn :
  string list list * {first:string, last:'a, middle:'b}
   -> {first:string, last:'a, middle:'b} list
datatype suit = Clubs | Diamonds | Hearts | Spades
datatype rank = Ace | Jack | King | Num of int | Queen
type card = suit * rank
datatype color = Black | Red
datatype move = Discard of suit * rank | Draw
exception IllegalMove
val card_color = fn : suit * 'a -> color
val card_value = fn : 'a * rank -> int
val remove_card = fn : ''a list * ''a * exn -> ''a list
val all_same_color = fn : (suit * 'a) list -> bool
val sum_cards = fn : ('a * rank) list -> int
val score = fn : (suit * rank) list * int -> int
val officiate = fn : (suit * rank) list * move list * int -> int

val it = () : unit

stdIn:17.20 Warning: calling polyEqual

val test = fn : string * ''a * ''a -> string * string
```

```
val it = ("all_except_option","Ok") : string * string

val it = ("all_except_option","Ok") : string * string

val it = ("all_except_option","Ok") : string * string

val it = ("get_substitutions1","Ok") : string * string

val it = ("get_substitutions1","Ok") : string * string

val it = ("get_substitutions1","Ok") : string * string

val it = ("get_substitutions2","Ok") : string * string

val it = ("get_substitutions2","Ok") : string * string

val it = ("get_substitutions2","Ok") : string * string

val it = ("similar_names","Ok") : string * string

val it = ("similar_names","Ok") : string * string

val it = ("similar_names","Ok") : string * string
```

```
val test_card1 = (Hearts,Jack) : suit * rank

val test_card2 = (Clubs,Num 8) : suit * rank

val test_card3 = (Diamonds,Ace) : suit * rank

val test_card_list1 = [(Hearts,Jack),(Clubs,Num 8),(Diamonds,Ace)] :
  (suit * rank) list

val test_card_list2 = [(Hearts,Jack),(Diamonds,Ace)] : (suit * rank) list

val test_card_list3 = [] : 'a list

val it = ("card_color","Ok") : string * string

val it = ("card_color","Ok") : string * string

val it = ("card_color","Ok") : string * string

val it = ("card_value","Ok") : string * string

val it = ("card_value","Ok") : string * string

val it = ("card_value","Ok") : string * string

val it = ("remove_card","Ok") : string * string


uncaught exception IllegalMove
  raised at: task.sml:102.33
-

uncaught exception IllegalMove
  raised at: task.sml:102.33
```

```
val it = ("all_same_color","Ok") : string * string

val it = ("all_same_color","Ok") : string * string

val it = ("all_same_color","Ok") : string * string

val it = ("sum_cards","Ok") : string * string

val it = ("sum_cards","Ok") : string * string

val it = ("sum_cards","Ok") : string * string

val it = ("score","Ok") : string * string

val it = ("score","Ok") : string * string

val it = ("score","Ok") : string * string

val it = ("officiate","Ok") : string * string

val it = ("officiate","Ok") : string * string

val it = ("officiate","Ok") : string * string
```

Всі функції пройшли всі підготовлені тести