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 №3

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

1. Завдання лабораторної роботи

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.

2. Опис програмного коду

task1.sml

```
use "hw02.sml";

    (*1.a*)
fun all_except_option(str, strlist) =
    let fun in_fun(strlist, resList, isFound) =
        case strlist of
            [] => (resList, isFound)
            |(hd::tl) => if (same_string(hd, str)) then
                            in_fun(tl, resList, true)
                        else
                            in_fun(tl, hd::resList, isFound)
    in
        let fun rev_list(lst, resList) =
            case lst of
                [] => resList
                | hd::tl => rev_list(tl, hd::resList)
        in
            case in_fun(strlist, [], false) of
                (hd::tl, true) => SOME(rev_list(hd::tl, []))
                |(list, false) => NONE
                |([], true) => SOME([])
        end
    end
;


    (*1.b*)
fun get_substitutions1(strListList, str) =
    case strListList of
        [] => []
        |hd::tl => case all_except_option(str, hd) of
                        SOME list => list @ get_substitutions1 (tl, str)
                        |NONE => get_substitutions1(tl, str)
;


    (*1.c*)
fun get_substitutions2(strListList, str) =
    let fun in_fun(curList, resList) =
        case curList of
            [] => resList
            | hd::tl => case all_except_option(str, hd) of
                            SOME list => in_fun(tl, resList @ list)
                            |NONE => in_fun(tl, resList)
```

```
    in
        in_fun(strListList, [])
    end
;


    (*1.d*)
fun similar_names(nameListList, {first = fname, middle = mname, last = lname}) =
    let fun in_fun(alternatFNameList, fullNamesList) =
        case alternatFNameList of
            [] => fullNamesList
            |hd :: tl => in_fun(tl,  {first = hd, middle = mname, last = lname}
:: fullNamesList)
    in
       {first = fname, middle = mname, last = lname} ::
in_fun(get_substitutions2(nameListList, fname), [])
    end
;
```

task2.sml

```
use "hw02.sml";


    (*2.a*)
fun card_color(card) =
    case card of
        (Clubs, _) => Black
        |(Spades, _) => Black
        |_ => Red
;


  (*2.b*)
fun card_value(card) =
    case card of
        (_, Num x) => x
        |(_, Ace) => 11
        |_ => 10
;


    (*2.c*)
fun remove_card (cs, c, e) =
    let fun in_fun (curCards, resCards) =
        case curCards of
            [] => raise e
            | hd::tl => if hd = c then
                            resCards @ tl
```

```
                        else
                            in_fun(tl, hd :: resCards)
    in
        in_fun(cs, [])
    end
;


    (*2.d*)
fun all_same_color (cards) =
    let val color = case cards of
                        [] => Black
                        |hd::tl => card_color(hd)
    fun in_fun(list, color) =
        case list of
            [] => true
            |hd::tl =>  if card_color(hd) = color then
                            in_fun(tl, color)
                        else
                            false
    in
        in_fun(cards, color)
    end
;


    (*2.e*)
fun sum_cards(cards) =
    let fun in_fun(cards, sum) =
        case cards of
            [] => sum
            | hd::tl => in_fun(tl, card_value(hd) + sum)
    in
        in_fun(cards, 0)
    end
;


    (*2.f*)
fun score(cards, goal) =
    let val preScore = if sum_cards(cards) > goal then
                            3 * (sum_cards(cards) - goal)
                       else
                            (goal - sum_cards(cards))
    in
        if all_same_color(cards) then
            preScore div 2
        else
            preScore
    end
;
```

```
    (*2.g*)
fun officiate(cards, moves, goal) =
    let fun next_move(handCards, curCards, curMoves) =
        if sum_cards(handCards) > goal then
            score(handCards, goal)
        else
            case curMoves of
                [] => score(handCards, goal)
                |hdMoves::tlMoves => case hdMoves of
                                        Discard card =>
next_move(remove_card(handCards, card, IllegalMove), curCards, tlMoves)
                                        |Draw => case curCards of
                                                    [] => score(handCards, goal)
                                                    | hdCard::tlCards =>
next_move(hdCard :: handCards, tlCards, tlMoves)
    in
        next_move([], cards, moves)
    end
;
```

hw02.sml (даний в умові задачі файл)

```
(* 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

(* put your solutions for problem 1 here *)

(* 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 *)
```

3. Тести програми

test1.sml

```
    (*test task 1 a*)
val test1_1a = all_except_option("ddd", ["3", "ddd", "1", "2"]);
val test2_1a = all_except_option("ddd", ["3", "ddd1", "1", "2"]);
val test3_1a = all_except_option("ddd", ["ddd"]);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_1a = SOME ["3","1","2"] : string list option

val test2_1a = NONE : string list option

val test3_1a = SOME [] : string list option
```

```
    (*test task 1 b*)
val test1_1b = get_substitutions1([],"Jack");
val test2_1b = get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Jeff");
val test3_1b = get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Olexandr");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_1b = [] : string list

val test2_1b = ["Jeffrey","Geoff","Jeffrey"] : string list

val test3_1b = [] : string list
```

```
    (*test task 1 c*)
val test1_1c = get_substitutions2([],"Jack");
val test2_1c = get_substitutions2([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Jeff");
val test3_1c = get_substitutions2([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Olexandr");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test1_1c = [] : string list

val test2_1c = ["Jeffrey","Geoff","Jeffrey"] : string list

val test3_1c = [] : string list
```

```
20
21      (*test task 1 d*)
22    val test1_1d = similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]],
23    {first="Fred", middle="W", last="Smith"});
24    val test2_1d = similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]],
25    {first="Olexandr", middle="W", last="Smith"});
26    val test3_1d = similar_names([],{first="Olexandr", middle="W", last="Smith"});
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_1d =
  [{first="Fred",last="Smith",middle="W"},{first="F",last="Smith",middle="W"},
   {first="Freddie",last="Smith",middle="W"},
   {first="Fredrick",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list

val test2_1d = [{first="Olexandr",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list

val test3_1d = [{first="Olexandr",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list
-
```

test2.sml

```
3       (*test task 2 a*)
4     val test1_2a = card_color((Hearts, Num 2));
5     val test2_2a = card_color((Clubs, Jack));
6     val test3_2a = card_color((Diamonds, Ace));
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test1_2a = Red : color

val test2_2a = Black : color

val test3_2a = Red : color
-
```

```
(*test task 2 b*)
val test1_2b = card_value((Hearts, Num 9));
val test2_2b = card_value((Clubs, Jack));
val test3_2b = card_value((Spades, Ace));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_2b = 9 : int

val test2_2b = 10 : int

val test3_2b = 11 : int
```

```
(*test task 2 c*)
val test1_2c = remove_card([(Hearts, Num 10), (Hearts, Num 9), (Diamonds, King), (Spades, Ace)],
(Spades, Ace), IllegalMove);
val test2_2c = remove_card([(Hearts, Num 9), (Hearts, Num 9), (Diamonds, King), (Spades, Ace)],
(Hearts, Num 9), IllegalMove);
val test3_2c = remove_card([(Hearts, Num 10), (Hearts, Num 9), (Diamonds, King), (Spades, Ace)],
(Hearts, Ace), IllegalMove);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_2c = [(Diamonds,King),(Hearts,Num 9),(Hearts,Num 10)] :
  (suit * rank) list

val test2_2c = [(Hearts,Num 9),(Diamonds,King),(Spades,Ace)] :
  (suit * rank) list


uncaught exception IllegalMove
  raised at: stdIn:97.25
```

```
(*test task 2 d*)
val test1_2d = all_same_color([(Diamonds, Num 10),(Diamonds, Jack),(Hearts, Ace)]);
val test2_2d = all_same_color([(Diamonds, Num 10),(Diamonds, Jack),(Spades, Ace)]);
val test3_2d = all_same_color([]);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1_2d = true : bool

val test2_2d = false : bool

val test3_2d = true : bool
```

```
29
30      (*test task 2 e*)
31    val test1_2e = sum_cards([]);
32    val test2_2e = sum_cards([(Diamonds, Num 10),(Diamonds, Jack),(Hearts, Ace)]);
33
34
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test1_2e = 0 : int

val test2_2e = 31 : int
-
```

```
35      (*test task 2 f*)
36    val test1_2f = score([(Diamonds, Num 10),(Diamonds, Jack),(Hearts, Ace)], 20);
37    val test2_2f = score([(Diamonds, Num 10),(Diamonds, Jack),(Hearts, Ace)], 34);
38    val test3_2f = score([(Diamonds, Num 10),(Spades, Jack),(Hearts, Ace)], 20);
39    val test4_2f = score([(Diamonds, Num 10),(Spades, Jack),(Hearts, Ace)], 34);
40
41
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test1_2f = 16 : int

val test2_2f = 1 : int

val test3_2f = 33 : int

val test4_2f = 3 : int
-
```

```
41
42      (*test task 2 g*)
43    val test1_2g = officiate([(Diamonds, Num 10),(Diamonds, Jack),(Hearts, Ace)],
44    [Draw, Draw],
45    20);
46    val test2_2g = officiate([(Diamonds, Num 10), (Spades, Jack), (Diamonds, Ace),(Hearts, Ace)],
47    [Draw, Draw, Discard (Spades, Jack), Draw],
48    21);
49    val test3_2g = officiate([(Diamonds, Num 10), (Spades, Ace), (Spades, Jack), (Hearts, Ace)],
50    [Draw, Draw, Discard (Diamonds, Ace)],
51    20);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test1_2g = 0 : int

val test2_2g = 0 : int

val test3_2g = 3 : int
-
```