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

Кафедра інформатики та програмної інженерії

Програмування інтелектуальних інформаційних систем

ЗВІТ

до лабораторних робіт

Виконав студент ІП-01 Смислов Даніл _____
(№ 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

```
(* TASK 1*)

(*a*)
fun same_string(s1 : string, s2 : string) =
    s1 = s2

fun all_except_option(str, strlst) =
   case strlst of
   [] => NONE
   |(hd::tl) => if same_string(hd,str) then SOME(tl)
                else case all_except_option(str,tl) of
                          NONE => NONE
                         |SOME exceptOption => SOME(hd::exceptOption)

(*b*)
fun get_substitutions1(strlstlst, str) =
   case strlstlst of
   [] => []
   |(hd::tl) => case all_except_option(str,hd) of
                NONE => get_substitutions1(tl,str)
                | SOME strlst => strlst @ get_substitutions1(tl,str);

(*c*)
fun get_substitutions2(strlstlst, str) =
   let fun temp(strlstlst,str,acc) =
      case strlstlst of
         [] => acc
         |hd::tl => case all_except_option(str,hd) of
                NONE => temp(tl,str,acc)
                | SOME strlst => temp(tl,str,acc @ strlst)
   in
      temp(strlstlst,str,[])
   end;

(*d*)
fun similar_names(strlstlst, {first = f, middle = m, last = l}) =
let fun temp(strlst) =
   case strlst of
      [] => []
      |(hd::tl) => {first = hd, middle = m, last = l} :: temp(tl)
in
   {first = f, middle = m, last = l} :: temp(get_substitutions1(strlstlst,f))
end;
```

```
(* TASK 2*)
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

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

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

(*c*)
fun remove_card(cs,c,e) =
   case cs of
   [] => raise IllegalMove
   |(hd::tl) => if hd = c then tl
                else hd::remove_card(tl,c,e);

(*d*)
fun all_same_color(cs) =
   case cs of
   [] => true
   |el::[] => true
   |(hd::md::tl) => if card_color(hd) = card_color(md) then
                                        all_same_color(md::tl)
                                        else false;
```

```sml
(*e*)
fun sum_cards(cs) =
   let fun temp(cs,acc)=
      case cs of
         []=> acc
         |(hd::tl) => temp(tl,acc + card_value(hd))
   in
      temp(cs,0)
   end;

(*f*)
fun score(cs,goal) =
let fun subScore(cs) =
   case sum_cards(cs)>goal of
   true =>  3*(sum_cards(cs)-goal)
   |false =>  goal - sum_cards(cs)
in
   case all_same_color(cs)of
   false => subScore(cs)
   |true => subScore(cs) div 2
end

(*g*)
fun officiate(cs,moves,goal)=
   let fun move(hand,deck,movesLst) =
   case sum_cards(hand) > goal
      of
      true => score(hand,goal)
      |false => case movesLst of
                  [] => score(hand,goal)
                  |(hd::tl) => case hd of
                                 Discard card => move(remove_card(hand,card,IllegalMove),deck,tl)
                                 |Draw => case deck of
                                             []=>score(hand,goal)
                                             | (hdDeck::tlDeck) => move(hdDeck::hand,tlDeck,tl)
   in
      move([],cs,moves)
   end;
```

3. Скріншоти тестування функцій

```sml
38   (*a*)
39   val test1 = all_except_option("check2",["check2","check1","check3"]) (*SOME ["check1","check3"]*)
40   val test2 = all_except_option("check",["check2","check1","check3"]) (*NONE*)
41   val test3 = all_except_option("check3",["check2","check1","check3"]) (*SOME ["check2","check1"]*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test1 = SOME ["check1","check3"] : string list option
val test2 = NONE : string list option
val test3 = SOME ["check2","check1"] : string list option
```

```sml
43   (*b*)
44   val test4 = get_substitutions1([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Jeff"); (*["Jeffrey","Geoff","Jeffrey"]*)
45   val test5 = get_substitutions1([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], "Fred") (*["Fredrick","Freddie","F"]*)
46   val test6 = get_substitutions1([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], "Petro") (*[]*)
47
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test4 = ["Jeffrey","Geoff","Jeffrey"] : string list

val test5 = ["Fredrick","Freddie","F"] : string list
val test6 = [] : string list
```

```
48  (*c*)
49  val test7 = get_substitutions2([["Fred","Fredrick"],["Jeff","Jeffrey"],["Geoff","Jeff","Jeffrey"]], "Jeff"); (*["Jeffrey","Geoff","Jeffrey"]*)
50  val test8 = get_substitutions2([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], "Fred") (*["Fredrick","Freddie","F"]*)
51  val test9 = get_substitutions2([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]], "Petro") (*[]*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test7 = ["Jeffrey","Geoff","Jeffrey"] : string list

val test8 = ["Fredrick","Freddie","F"] : string list
val test9 = [] : string list
```

```
53  (*d*)
54  val test10 = similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]],
55  {first="Fred", middle="W", last="Smith"});
56  (*[{first="Fred",last="Smith",middle="W"},
57    {first="Fredrick",last="Smith",middle="W"},
58    {first="Freddie",last="Smith",middle="W"},
59    {first="F",last="Smith",middle="W"}]*)
60  val test11 = similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]],
61  {first="Betty", middle="W", last="Smith"});
62  (*[{first="Betty",last="Smith",middle="W"},
63    {first="Elizabeth",last="Smith",middle="W"}]*)
64  val test12 = similar_names([["Fred","Fredrick"],["Elizabeth","Betty"],["Freddie","Fred","F"]],
65  {first="Stas", middle="W", last="Smith"});
66  (*[{first="Stas",last="Smith",middle="W"}]*)
67
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-
val test10 =
  [{first="Fred",last="Smith",middle="W"},
   {first="Fredrick",last="Smith",middle="W"},
   {first="Freddie",last="Smith",middle="W"},
   {first="F",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list

val test11 =
  [{first="Betty",last="Smith",middle="W"},
   {first="Elizabeth",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list

val test12 = [{first="Stas",last="Smith",middle="W"}] :
  {first:string, last:string, middle:string} list
```

```
69  (*Task 2*)
70
71  (*a*)
72  val test13 = card_color((Diamonds,Jack)); (*Red*)
73  val test14 = card_color((Hearts,Num 8)); (*Red*)
74  val test15 = card_color((Spades,Ace)); (*Black*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test13 = Red : color

val test14 = Red : color

val test15 = Black : color
```

```
76   (*b*)
77   val test16 = card_value((Diamonds,Ace)); (*11*)
78   val test17 = card_value((Diamonds,Jack)); (*10*)
79   val test18 = card_value((Spades,Num 9)); (*9*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test16 = 11 : int

val test17 = 10 : int

val test18 = 9 : int
```

```
81   (*c*)
82   val test19 = remove_card([(Hearts,King)],(Hearts,King),IllegalMove); (*[]*)
83   val test20 = remove_card([(Hearts,King),(Spades,King)],(Spades,King),IllegalMove); (*[(Hearts,King)]*)
84   val test21 = remove_card([(Hearts,Queen),(Spades,Num 10), (Hearts,Ace)],(Spades,Num 10),IllegalMove); (*[(Hearts,Queen),(Hearts,Ace)]*
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test19 = [] : (suit * rank) list

val test20 = [(Hearts,King)] : (suit * rank) list

val test21 = [(Hearts,Queen),(Hearts,Ace)] : (suit * rank) list
```

```
86   (*d*)
87   val test22 = all_same_color([(Hearts,King)]); (*true*)
88   val test23 = all_same_color([]); (*true*)
89   val test24 = all_same_color([(Hearts,King),(Clubs,Ace)]); (*false*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test22 = true : bool

val test23 = true : bool

val test24 = false : bool
```

```
91   (*e*)
92   val test25 = sum_cards([(Hearts,King),(Clubs,Num 5)]); (*15*)
93   val test26 = sum_cards([]); (*0*)
94   val test27 = sum_cards([(Hearts,King),(Clubs,Num 5),(Clubs,Ace)]); (*26*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test25 = 15 : int

val test26 = 0 : int

val test27 = 26 : int
```

```
96    (*f*)
97    val test28 = score([(Hearts,King),(Clubs,Num 5)],10); (*15*)
98    val test29 = score([(Diamonds,Jack),(Spades,Ace)],21); (*0*)
99    val test30 = score([(Hearts,Ace),(Clubs,Num 10), (Diamonds,Queen)],25); (*18*)
100
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test28 = 15 : int

val test29 = 0 : int

val test30 = 18 : int
```

```
101   (*g*)
102   val test31 = officiate([(Diamonds,Jack),(Spades,Ace)],[Draw,Draw],21); (*0*)
103   val test32 = officiate([(Diamonds,Jack),(Spades,Ace),(Clubs, Num 6)],[Draw,Draw,Discard (Diamonds, Jack),Draw],21); (*2*)
104   val test33 = officiate([(Diamonds,Jack),(Spades,Ace)],[Draw,Draw,Discard(Spades, Num 7)],20); (*3*)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
val test31 = 0 : int

val test32 = 2 : int

val test33 = 3 : int
```