CMSC330 Fall 2024 Quiz 2 Solutions

Problem 1: Basics

[Total 4 pts]

	True	False
Regular expressions can describe strings with 2 sets of balanced parenthesis Because there is exactly 2, you can make sure its followed	(T)	(F)
In the expression `let x = ref 4 in let y = x in !y`, both x and y point to the same thing x points to 4, and y also points to 4. if you change x, y also changes and vice versa	(T)	(F)
The **concept** of fold is limited to lists you can fold over many data structures, lists, trees, custom ones	(T)	(F)
The variables f in `let f () = print_int 3` and `let f = print_int 3` have the same behavior when used f () a function so 3 will be printed every time you call it. The second, f is the result of printing 3 which is unit.	(T)	(F)
Regular expressions cannot describe arbitrarily sized palindromes	(T)	(F)
In the expression `let x = ref 4 in let y = x in !y`, x and y point to different memory addresses	(T)	(F)
Regular expressions can describe palindromes of size 3	(T)	(F)
You can write `map` in terms of `fold_left / fold_right` (ignoring side effects/unit operations)	(T)	(F)
`let f () = print_int 3` and `let f = print_int 3` have the same behavior when using `f`	(T)	(F)
Regular expressions can describe strings with an arbitrary number of balanced parenthesis	(T)	(F)

Problem 2: Data types and Map

[Total 6 pts]

Consider the following Variant:

```
type 'a tree = Leaf|BiNode of 'a * 'a tree * 'a tree (* value, left subtree, right subtree *)
```

Suppose we have a function called `tree map`. It works like `map`, but will map a `'a tree` to `'b tree`.

Using only `tree_map` and `fold_left`, write a function `even_sums` that takes in an `int list tree` and returns a `bool tree`. Each node in the output tree should represent if the sum of the input node is even.

You can write additional helper functions, **but may not use the `rec` keyword**

```
val tree_map f t: ('a -> 'b) -> 'a tree -> 'b tree
val fold_left f a l: ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a

even_sums (BiNode([], BiNode([1;3;5], Leaf, Leaf), BiNode([2;1;7], Leaf, Leaf)))
 => (BiNode(true, BiNode(false, Leaf, Leaf), BiNode(true, Leaf, Leaf)))
        []                      true
       /   \          =>       /    \
[1;3;5]    [2;1;7]          false  true
```

```
let even_sums t = tree_map (fun x -> (fold_left (+) 0 x) mod 2 = 0) t
```

Using only `tree_map` and `fold_left`, write a function `odd_sums` that takes in an `int list tree` and returns a `bool tree`. Each node in the output tree should represent if the sum of the input node is odd.

```
odd_sums (BiNode([], BiNode([1;3;5], Leaf, Leaf), BiNode([2;1;7], Leaf, Leaf)))
 => (BiNode(false, BiNode(true, Leaf, Leaf), BiNode(false, Leaf, Leaf)))
        []                      false
       /   \          =>       /    \
[1;3;5]    [2;1;7]          true  false
```

```
let odd_sums t = tree_map (fun x -> (fold_left (+) 0 x) mod 2 <> 0) t
```

Problem 3: Regex

[Total 6 pts]

Write a regex that describes **exactly** the room names found in CS related buildings. A room name will have

- the building code (**only** IRB or AVW)
- followed by the room number (any 4 digit number from 0000 to 4500 (inclusive))
- followed by a space and the purpose which is either the last name of the professor or "TA SPACE")
- Last names of all professors will **begin** with any capital letter and have **at least** 3 lowercase letters following their first letter

valid room names	**invalid room names**
`IRB2238 Baka`	`HJP1206This`
`IRB4500 Mamat`	`IRB2248 bakalian`
`AVW4165 TA SPACE`	`avw123 Bad`

```
^(IRB|AVW)(([0-3]\d{3})|(4([0-4]\d{2}|500))) ((TA SPACE)|([A-Z][a-z]{3,}))$
```

Write a regex that describes **exactly** the room names found in CS related buildings. A room name will have

- the building code (**only** IRB or CSI)
- followed by the room number (any 4 digit number from 4500 to 8000 (inclusive))
- followed by a space and the purpose which is either the last name of the professor or "TA SPACE"
- Last names of professors will begin with a capital letter and have **at least** 3 lowercase letters following their first letter

valid room names	**invalid room names**
`IRB5400 Baka`	`HJP1206This`
`IRB4510 Mamat`	`IRB2248 bakalian`
`CSI8000 TA SPACE`	`avw123 Bad`

```
^(IRB|CSI)((4[5-9]\d{2})|([5-7]\d{3}|8000)) ((TA SPACE) | ([A-Z][a-z]{3,}))$
```

Write a regex that describes **exactly** the room names found in CS related buildings. A room name will have

- the building code (**only** CSI or AVW)
- followed by the room number (any 4 digit number from 4500 to 8000 (inclusive))
- followed by a space and the purpose (Either the last name of the professor or "TA SPACE")
- Last names of all professors will **begin** with any capital letter and have **at least** 3 lowercase letters following their first letter

valid room names	**invalid room names**
`CSI5400 Baka`	`HJP1206This`
`CSI4510 Mamat`	`IRB2248 bakalian`
`AVW8000 TA SPACE`	`csi123 Bad`

```
^(CSI|AVW)((4[5-9]\d{2})|([5-7]\d{3}|8000)) ((TA SPACE)|([A-Z][a-z]{3,}))$
```

Problem 4: Property Based Testing

[Total 4 pts]

Consider the following function which has a bug in it:

```
1 (* signed_square should take in an int x, square it, but keep the original sign *)
2 let signed_square x = if x < 0 then (-x) * (-x) else x * x
```

Consider the following property: *the output of `signed_square` should be greater than or equal to the input*

Is this a valid property? Yes/No: (Y) (N)

Is the function `fun x -> signed_square x >= x` a correct representation of the property? Yes/No: (Y) (N)

If we test this property on the provided code, will it ever return false?

Yes: (Y) The property is not valid so the result of testing this property is meaningless: (NA)

No: (N)

Consider the following property: *the parity of the output should match the parity of the input*

Is this a valid property? Yes/No: (Y) (N)

Is the function
`fun x -> if (signed_square x) mod 2 = 0 then x mod 2 = 0 else x mod 2 <> 0`
a correct representation of the property? Yes/No: (Y) (N)

If we test this property on the provided code, will it ever return false?

Yes: (Y) The property is not valid so the result of testing this property is meaningless: (NA)

No: (N)