



CMSC330 Spring 2025 Quiz 2 Solutions

Problem 1: Basics

[Total 4 pts]

	True	False
$\exists$ a regular expression that describes strings of any length that contain any number of balanced parentheses	☐	☒
Property Based Testing is intended to completely replace unit testing	☐	☒
The expression <code>(fun x y -> 3) (print_string "a") (print_string "b")</code> always prints out "ab"	☐	☒
Regular expressions can only describe a finite set of strings	☐	☒
No regular expression can describe strings of any length that contain any number of balanced parentheses	☒	☐

Problem 2: Property Based Testing

[Total 6 pts]

Consider the following incorrect `tree_map` function.

```
type tree = Leaf of int | Node of int * tree * tree
```

```
(* has bug(s)! *)
let rec tree_map f tree = match tree with
  Leaf(x) -> Leaf(x)
  | Node(x,l,r) -> Node(f x, tree_map f l, r)
```

Consider the following property p about the `tree_map` function:

p : If calling `tree_map` on some tree $t1$ results in tree $t2$, then $t1$'s root value should be different than $t2$'s root value

Using a correct implementation of `tree_map`, this property p should hold true for all valid inputs?

☐ Yes ☒ No

Using our implementation of `tree_map`, this property p would **not** hold true for all valid inputs?

☒ Yes ☐ No

Suppose I encode this property in OCaml to be used in OCaml's QCheck library as the following:

```
let prop f tree = match (tree, tree_map f tree) with
  Leaf(x), Leaf(x) -> true
  | Node(x, -, -), Node(y, -, -) -> if x <> y then true else false
  | _, _ -> false
```

The above `prop` function is a valid encoding of the property p .

☐ Yes ☒ No

```
type tree = Leaf of int | Node of int * tree * tree
```

```
(* has bug(s)! *)  
let rec tree_map f tree = match tree with  
  Leaf(x) -> Leaf(f x)  
  |Node(x,l,r) -> Node(x, tree_map f l, tree_map f r)
```

Consider the following property p about the `tree_map` function:

p : `tree_map` should not change the number of leaves

Using a correct implementation of `tree_map`, this property p should hold true for all valid inputs?

☒ Yes ☐ No

Using our implementation of `tree_map`, this property p would **not** hold true for all valid inputs?

☐ Yes ☒ No

Suppose I encode this property in OCaml to be used in OCaml's QCheck library as the following:

```
let prop f tree = count_leaves tree = count_leaves tree_map tree
```

The above prop function is a valid encoding of the property p . ☐ Yes ☒ No

type tree = Leaf of int | Node of int * tree * tree

```
(* has bug(s)! *)  
let rec tree_map f tree = match tree with  
  Leaf(x) -> Leaf(x)  
  |Node(x,l,r) -> Node(f x, l, r)
```

Consider the following property p about the `tree_map` function:

p : calling `tree_map` using the identity function should not change the tree

Using a correct implementation of `tree_map`, this property p should hold true for all valid inputs?

☒ Yes ☐ No

Using our implementation of `tree_map`, this property p would **not** hold true for all valid inputs?

☐ Yes ☒ No

Suppose I encode this property in OCaml to be used in OCaml's QCheck library as the following:

```
let prop f tree = tree_map (fun x -> x) tree = tree
```

The above prop function is a valid encoding of the property p . ☒ Yes ☐ No

```
type tree = Leaf of int | Node of int * tree * tree
```

```
(* has bug(s)! *)  
let rec tree_map f tree = match tree with  
  Leaf(x) -> Leaf(x)  
  |Node(x,l,r) -> Node(f x, tree_map f r, tree_map f l)
```

Consider the following property p about the `tree_map` function:

p : `tree_map` should not change the shape of the tree

Using a correct implementation of `tree_map`, this property p should hold true for all valid inputs?

☒ Yes ☐ No

Using our implementation of `tree_map`, this property p would **not** hold true for all valid inputs?

☒ Yes ☐ No

Suppose I encode this property in OCaml to be used in OCaml's QCheck library as the following:

```
let prop f tree = same_shape (tree_map tree) tree
```

Assuming there exists a function called `same_shape` which returns true if the two trees as input are the same shape and false otherwise, the above prop function is a valid encoding of the property p . ☐ Yes ☒ No

Problem 3: Regex

[Total 10 pts]

*	zero or more repetitions of the preceding character or group
+	one or more repetitions of the preceding character or group
?	zero or one repetitions of the preceding character or group
.	any character
$r_1 r_2$	r_1 or r_2 (eg. $a b$ means 'a' or 'b')
[abc]	match any character in abc
[$\sim r_1$]	anything except r_1 (eg. [$\sim abc$] is anything but an 'a', 'b', or 'c')
[r_1-r_2]	range specification (eg. [a-z] means any letter in the ASCII range of a-z)
{n}	exactly n repetitions of the preceding character or group
{n,}	at least n repetitions of the preceding character or group
{m,n}	at least m and at most n repetitions of the preceding character or group
^	start of string
\$	end of string

(a) Names and Ages

[4 pts]

Write a regex that describes a person's name and their age in the format:

Name: age

- A Name starts with a Capital Letter followed by any number (0 or more) of lowercase letters
- An age is a valid integer from 1 - 100 (can be 0 padded)

Valid Examples of Names and Ages names

Invalid examples of Names and Ages names

Baka: 42
Mamat: 24
W: 002

nocaptial: -34
Hasnumber6: 200
Nointeger: 3.14

`^[A-Z][a-z]*: 0*([1-9][0-9]?|100)`

Write a regex that describes a person's name and their age in the format:

Name: age

- A Name starts with 4 Capital Letters followed by any number (0 or more) of lowercase letters
- An age is a valid integer from 1000 - 1100 (can be 0 padded)

Valid Examples of Names and Ages names

Invalid examples of Names and Ages names

BAKA: 1042
MAMAt: 1024
SUSSman: 001002

nocaptial: -34
Hasnumber6: 1200
Nointeger: 1003.14

`^[A-Z]{4}[a-z]*: 0*(10[0-9]{2} | 1100)`

Write a regex that describes a person's name and their age in the format:

Name: age

- A Name starts with two Capital Letters followed by any number of lowercase letters
- An age is a valid integer from 50 - 92(inclusive) (can be 0 padded)

Valid Examples of Names and Ages names

BAka: 84
MAmat: 92
WAluiji: 0050

Invalid examples of Names and Ages names

nocaptial: -34
Hasnumber6: 200
N0integer: 63.14

`^[A-Z]{2}[a-z]*: 0*([5-8][0-9] | 9[0-2])`

Write a regex that describes a person's name and their age in the format:

Name: age

- A Name starts with a lowercase letter followed by any number of uppercase letters
- An age is a valid integer from 200 - 300 (inclusive) (can be 0 padded)

Valid Examples of Names and Ages names

bAKA: 222
mAMAT: 234
wARIO: 00200

Invalid examples of Names and Ages names

NOLOWER: -34
hASNUMBER5: 200
noFLOATS: 3.14

`^[a-z][A-Z]*: 0*(2[0-9]{2} | 300)`

(b) Addresses

[6 pts]

Write a regex that describes **exactly** a youtube video URL.

- Will always start with: youtube.com/watch?
- Will be followed with 1 or more key-value pairs in the form: key=value
- Keys will be single lowercase letter
- Values will be any alphanumeric (lowercase, uppercase, digits), each at least 1 character long
- key-value pairs will be separated by an ampersand (&) character if there is more than 1 pair

valid urls

youtube.com/watch?v=dQw4w9WgXcQ
youtube.com/watch?v=XqZsoesa55w&t=10
youtube.com/watch?a=0&v=k85mRPqvMbE

invalid urls

youtube.com/watch?autoplay=1&video=djymZspawFc
youtube.com/watch?
youtube.com/watch?t=20v=Zq1QJ2QztgM

youtube\.com/watch\?

`([a-z]=([A-Z]|[a-z][0-9])*)&)*[a-z]=([A-Z]|[a-z][0-9])+`