Your submission to Mock Exam II: Functional Programming and Verification (Summer Semester 2022) (Ruan Viljoen)

Exam Mode: Test Exam

Date: Aug 10, 2022 Working Time: 2h 0min 0s Used working time: : 55min 3s (45.88 %)

Exercises: 6 Points: 60

Review is open

Exercise 1

Score 40%, 12 of 30 points (a few seconds ago)

Quiz [30 Points]

1) Weaker Statement

Which of the following statements is *weaker* than the statement B for arbitrary logical statements A, B?

Please choose the correct answer option

Your Score: 0/3

Answer		Solution		You
false	Explanation	Wrong	⚠	◉
$A \wedge B$	Explanation	Wrong		○
$\neg B$	Explanation	Wrong		○
$A \implies B$	Explanation	Correct	⚠	○

2) Stronger Statement

Which of the following statements is *stronger* than the statement $y \leq x + 15 \vee y = 12$ for all $x, y \in \mathbb{Z}$?

Please choose the correct answer option

Your Score: 3/3

Answer		Solution	You
$y < x \vee (y = 12 \wedge x \neq 5)$	Explanation	Correct	◉
$y \leq x + 15 \vee y = 15$	Explanation	Wrong	○
$y \leq \lvert x \rvert + 12$	Explanation	Wrong	○
true	Explanation	Wrong	○

3) Implications

Choose the statement that holds for all $x, y \in \mathbb{Z}$.

Please choose the correct answer option

Your Score: 0/3

Answer		Solution		You
$x \leq 5 \wedge x^2 \geq 36 \implies x < 0$	Explanation	Correct	⚠	○
$x > 1 \implies x \neq 0 \wedge y = 0$	Explanation	Wrong		○
$x \neq y \vee x = 0 \implies y = 0 \vee x = 0$	Explanation	Wrong		○
$y \in \{0, 1, 2\} \wedge x > y \implies y \in \{0, 1\} \wedge x \geq y$	Explanation	Wrong	⚠	◉

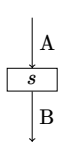
4) Weakest Preconditions

Given:

$A \equiv (a = 3 \wedge b = 1) \vee (a + b = 2 \wedge b = -1)$

$B \equiv a = 3 \wedge b < 0$

For which MiniJava statement s would the annotations on this node be locally consistent:



Please choose the correct answer option

Your Score: 0/3

Answer		Solution	You
`b = b - 2;`	Explanation	Correct	○
`;`	Explanation	Wrong	◉
`a = 0;`	Explanation	Wrong	○
`b = read();`	Explanation	Wrong	○

5) Pattern Matching

Given the definition of the type `point`:

```
type point = Point of int * int (* X-coordinate, Y-coordinate *)
```

Which of the following defines a valid OCaml function that takes a point (as defined above) as an argument and returns its X-coordinate?

Please choose the correct answer option

Your Score: 3/3

Answer		Solution	You
`let f p = match p with Point (x, y) -> x`	Explanation	Correct	◉
`let f p = match p with (x, y) -> x`	Explanation	Wrong	○
`let f p = match p with { x; y } -> x`	Explanation	Wrong	○
`let f p = match p with Point of (x * y) -> x`	Explanation	Wrong	○

6) Tail Recursion

Which of these functions is tail recursive according to the lecture and uses constant stack space when executed on arbitrary inputs?

Please choose the correct answer option

Your Score: 0/3

Answer		Solution	You
`let f xs ys = let rec helper xs ys acc = match xs, ys with | [], [] -> List.rev acc | c::cs, l::ls -> helper cs ls (l::c::acc) | _, _ -> [] in helper xs ys []`	Explanation	Correct	○
`let rec f ls n = match ls with | [] -> n+1 | x::xs -> f (List.rev xs) (n-x) + n`	Explanation	Wrong	◉
`let f ls = let rec helper xs acc = match xs with | [] -> acc | c::cs -> if c > 0 then helper cs (acc@[c]) else helper cs acc in helper ls []`	Explanation	Wrong	○
`let rec f ls = match ls with | [] -> [] | (b,x)::xs -> if b then f xs else x::f xs`	Explanation	Wrong	○

7) Generalization

Which of the given statements is correct and a useful generalization of the statement `map2 f xs [] = map f xs`?

```
let rec map2 f xs ys = match xs with
    | [] -> ys
    | l::ls -> map2 f ls (ys @ [f l])

let rec map f xs = match xs with
  | [] -> []
  | a::l -> f a :: map f l
```

Please choose the correct answer option

Your Score: 0/3

Answer		Solution	You
`map2 f xs ys = ys @ map f xs`	Explanation	Correct	○
`map2 f xs ys = (map f xs) @ ys`	Explanation	Wrong	◉
`n + map2 f xs [] = n + map f xs`	Explanation	Wrong	○

`map2 f xs ys = xs @ map f ys`	Explanation	Wrong	○

8) Equational Reasoning Tree

The following function and type definitions are given:

```
type tree = Node of tree * tree | Empty

let rec switch t = match t with
    | Empty -> Empty
    | Node (l, r) -> Node (switch r, switch l)
```

Additionally this induction hypothesis is given:

```
IH: switch (switch t) = t
```

Which of the following steps is a valid **single** equational reasoning step? You can assume that it is part of an inductive proof where you currentyl assume that the `IH` given above holds for `tree`s `a` and `b`.

Please choose the correct answer option

Your Score: 0/3

Answer		Solution	You
`Node (switch (switch a), switch (switch b)) (by IH) === Node (a, switch (switch b))`	Explanation	Correct	⚠ ○
`switch Empty (by switch) === Empty`	Explanation	Wrong	○
`Node (switch b, switch a) (by IH) === switch (Node (b, a))`	Explanation	Wrong	⚠ ◉
`switch (switch (Node (a, b))) (by IH) === Node (a, b)`	Explanation	Wrong	○

9) Equational Reasoning

The following function definitions are given:

```
let rec repeat n f x = match n with
    | 0 -> x
    | z -> repeat (z-1) f (f x)

let mul a b = a * b
let f x y = match x+y with | 0 -> -1 | n -> n
```

Which of the following steps is a valid **single** equational reasoning step? You can assume that it is part of an inductive proof.

Please choose the correct answer option

Your Score: 3/3

Answer		Solution	You
`repeat (5-1) (fun x -> x+1) ((fun x -> x+1) 2) (by fun) === repeat (5-1) (fun x -> x+1) (2+1)`	Explanation	Correct	◉
`f 2 3 (by f) === match 5 with | 0 -> -1 | n -> n`	Explanation	Wrong	○
`repeat (5-1) (fun x -> x+1) 8 (by repeat) === match (5-1) with | 0 -> 8 | z -> repeat (z-1) (fun x -> x+1) (fun x -> 8+1)`	Explanation	Wrong	○
`mul (5+0) (4+0) (by mul) === 5 * 4`	Explanation	Wrong	○

10) Implications

Choose the statement that holds for all $x, y \in \mathbb{Z}$.

Please choose the correct answer option

Your Score: 3/3

Answer		Solution	You
$x \leq 0 \wedge y * x > 0 \implies 0 \geq y$	Explanation	Correct	◉
$x \geq y \vee y \geq 0 \implies x^2 \geq y^2$	Explanation	Wrong	○
$x \geq 1 \wedge \lvert y \rvert \geq 0 \implies x \geq y$	Explanation	Wrong	○
$x < 0 \vee y > 0 \implies x \neq 0 \wedge y \neq 0$	Explanation	Wrong	○

Exercise 2

Score 33.3%, 2 of 6 points (a few seconds ago)

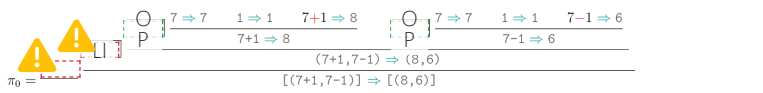
Big Step [6 Points]

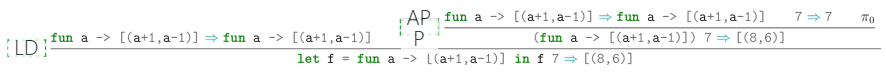
1) Big Rules

Annotate the missing rules in this Big-Step Tree.

Your Score: 2/6

Your Submission:





Exercise 3

No graded result

What Is the Point? [6 Points]

What's the point?

In this assignment, you are supposted to implement some functionalities to compute with euclidean vectors.

1. ? **string_of_vector3** No results
   Implement a function `string_of_vector3 : vector3 -> string` to convert a vector into a human-readable representation.
   The string for the zero vector should be: `(0.,0.,0.)`.
   Hint: use `string_of_float` to convert components.
2. ? **vector3_add** No results
   Write a function `vector3_add : vector3 -> vector3 -> vector3` that adds two vectors component-wise.
3. ? **vector3_max** No results
   Write a function `vector3_max : vector3 -> vector3 -> vector3` that returns the vector with the greater magnitude (i.e. the greater euclidean norm).
4. ? **combine** No results
   Write a function `combine : vector3 -> vector3 -> vector3 -> string` that adds its first argument to the larger of the other two arguments and returns the result as a string.

Submission Format

Make sure to commit and push your submission before the end of the exam.

You didn't submit any solution for this exercise.

Exercise 4

No graded result

Equational Reasoning [6 Points]

Equational Reasoning

The following functions are defined:

```
let rec nlen n l = match l with [] -> 0
  | h::t -> n + nlen n t

let rec fold_left f a l = match l with [] -> a
  | h::t -> fold_left f (f a h) t

let rec map f l = match l with [] -> []
  | h::t -> f h :: map f t

let (+) a b = a + b
```

Show that the statement

`nlen n l = fold_left (+) 0 (map (fun _ -> n) l)`

holds for arbitrary `l` and `n`. Assume that all expressions do terminate.

Submission Format

Submissions may only be in the form of plain text.

Your submission must following the following format. Copy this template into your submission and then complete it. Leave any fields you don't need blank.

```
Generalized statement (*) (if necessary): <...>
---
Base Case:
Statement being proven in base case: <...>
Proof of base case:
<...>
---
Inductive Step:
Induction hypothesis (or hypotheses): <...>
Statement being proved in inductive step: <...>
Proof of inductive step:
<...>
---
Instantiation of generalization (if necessary):
<...>
---
QED
```

For all equational proofs that show the equivalence of two MiniOCaml expressions, annotate each step as follows:

```
           e_1
(rule 1) = e_2
(rule 2) = e_3
     ...
(rule n) = e_n
```

For each step, when you:

- apply the definition of a function `f`, `rule` must be `f`
- apply the rule for function application, `rule` must be `fun`
- apply an induction hypothesis, `rule` must be `I.H.`
- simplify an arithmetic expression, `rule` must be `arith`
- select a branch in a match expression, `rule` must be `match`
- expand a `let` defintion, `rule` must be `let`
- apply a lemma that you have already proven in the exercise, `rule` must be the name you gave to the lemma

In each step, apply only a single rule. Write each step on its own line.

Points may be deducted for submissions that do not use the template or do not follow this format for equational proofs.

No submission

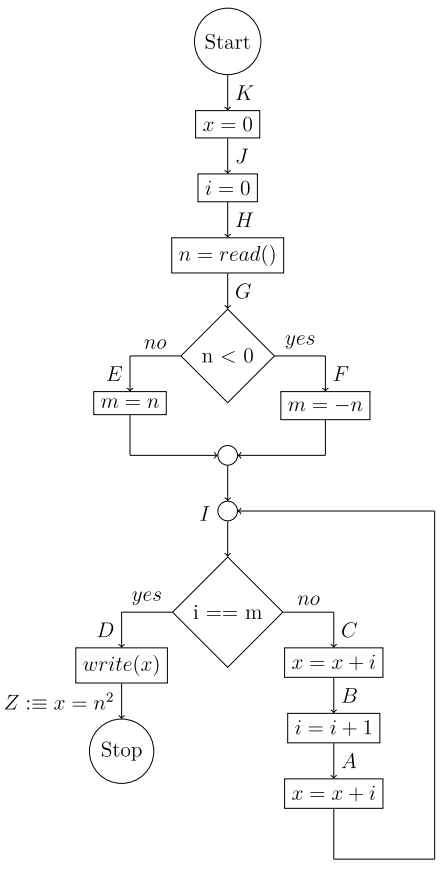
A Exercise 5

No graded result

Weakest Precondition [6 Points]

Weakest Precondition

Given is the following control flow graph:



Prove that Z holds using weakest preconditions.

Submission Format

Use the provided labels to refer to the program points.

For each application of the WP-operator list the inital result of the application and then simplify the expression by writing each step in its own line.

If your answer requires mathematical symbols, copy and paste the Unicode characters from the following table or type the ASCII-only variant yourself.

Symbol	Unicode	ASCII-only
Logical OR	∨	`OR`
Logical AND	∧	`AND`
Logical NOT	¬	`~`
Logical Implication	⟹	`==>`
Reversed Logical Implication	⟸	`<==`
Logically Equivalent	≡	`===`

Symbol	Unicode	ASCII-only
Universal Quantifier ("for all")	∀	`forall`
Existential Quantifier ("there exists")	∃	`exists`
Less-Than	<	`<`
Greater-Than	>	`>`
Less-Than or Equal To	≤	`<=`
Greater-Than or Equal To	≥	`>=`
Not Equal To	≠	`!=`
Set Membership ("in")	∈	`in`
Exponents	² ³	`^2` `^3` `^(a + b)`
Square Root	√	`sqrt(x)`
Weakest Precondition	WP⟦⟧( )	`WP[[ ]]( )`
Pi	π	`pi`
Tau	τ	`tau`

No submission

A Exercise 6

No graded result

Big Step [6 Points]

Big Step

Prove that the function

```
let rec mul = fun a b ->
  match a with 0 -> 0 | _ -> b + mul (a-1) b
```

terminates for all inputs $a, b \geq 0$, by filling the holes `<hole n>` in the following Big-Step proof:

Definitions

$$\pi_{mul} = \texttt{<hole 1>}\ \frac{\texttt{mul} = \texttt{fun a b -> match a with 0 -> 0 | \_ -> b + mul (a-1) b} \quad \texttt{fun a b -> match a with 0 -> 0 | \_ -> b + mul (a-1) b} \Rightarrow \texttt{fun a b -> match a with 0 -> 0 |}}{\texttt{mul} \Rightarrow \texttt{fun a b -> match a with 0 -> 0 | \_ -> b + mul (a-1) b}}$$

We prove by induction on `a` that `mul a b` terminates with $a * b$:

- **Base case: `<hole 2>`:**

$$\texttt{<hole 3>}\ \frac{\texttt{<hole 4>} \quad \texttt{<hole 5>}\ \dfrac{}{\texttt{<hole 6>} \Rightarrow \texttt{<hole 7>}}}{\texttt{mul 0 b} \Rightarrow 0}$$

- **Inductive case: Assume `mul a b` terminates for an $a \geq 0$. Now, we show that it also terminates for $a + 1$:**

$$\text{APP}\ \frac{\texttt{<hole 8>} \quad \text{PM}\ \dfrac{\text{OP}\ \dfrac{\text{APP}\ \dfrac{\texttt{<hole 9>}}{\texttt{mul (a+1-1) b} \Rightarrow a * b}\ b\ +\ (a * b) \Rightarrow (a+1) * b}{\texttt{b + mul (a+1-1) b} \Rightarrow (a+1) * b}}{\texttt{match a+1 with 0 -> 0 | \_ -> b + mul (a+1-1) b} \Rightarrow (a+1) * b}}{\texttt{mul (a+1) b} \Rightarrow (a+1) * b}$$

Submission Format

Write the content of one hole per line in the format:

```
hole 1 = <content 1>
hole 2 = <content 2>
       ⋮
```

If your answer requires mathematical symbols, copy and paste the Unicode characters from the following table or type the ASCII-only variant yourself.

Symbol	Unicode	ASCII-only
Logical OR	∨	`OR`
Logical AND	∧	`AND`
Logical NOT	¬	`~`
Logical Implication	⟹	`==>`
Reversed Logical Implication	⟸	`<==`
Logically Equivalent	≡	`===`
Universal Quantifier ("for all")	∀	`forall`
Existential Quantifier ("there exists")	∃	`exists`
Less-Than	<	`<`
Greater-Than	>	`>`
Less-Than or Equal To	≤	`<=`
Greater-Than or Equal To	≥	`>=`
Not Equal To	≠	`!=`
Set Membership ("in")	∈	`in`
Exponents	² ³	`^2` `^3` `^(a + b)`
Square Root	√	`sqrt(x)`

Symbol	Unicode	ASCII-only
Weakest Precondition	WP⟦ ⟧( )	`WP[[ ]]( )`
Pi	π	`pi`
Tau	τ	`tau`

No submission