Office Use Only

MONASH University

Semester Two 2017
Examination Period

Faculty of Information Technology

EXAM CODES: **FIT2102**

TITLE OF PAPER: **Programming Paradigms – Paper 1**

EXAM DURATION: 2 hours writing time

READING TIME: 10 minutes

THIS PAPER IS FOR STUDENTS STUDYING AT: (tick where applicable)

☐ Berwick ☑ Clayton ☑ Malaysia ☐ Off Campus Learning ☐ Open Learning
☐ Caulfield ☐ Gippsland ☐ Peninsula ☐ Monash Extension ☐ Sth Africa
☐ Parkville ☐ Other (specify)

During an exam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, mobile phone, smart watch/device, calculator, pencil case, or writing on any part of your body. Any authorised items are listed below. Items/materials on your desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession.

No examination materials are to be removed from the room. This includes retaining, copying, memorising or noting down content of exam material for personal use or to share with any other person by any means following your exam.
Failure to comply with the above instructions, or attempting to cheat or cheating in an exam is a discipline offence under Part 7 of the Monash University (Council) Regulations.

<u>**AUTHORISED MATERIALS**</u>

OPEN BOOK ☐ **YES** ☑ **NO**

CALCULATORS ☐ **YES** ☑ **NO**

SPECIFICALLY PERMITTED ITEMS ☐ **YES** ☑ **NO**
if yes, items permitted are:

Candidates must complete this section if required to write answers within this paper

STUDENT ID: __ __ __ __ __ __ __ __ DESK NUMBER: __ __ __ __ __

INSTRUCTIONS TO CANDIDATES:

- This paper contains ten (10) multiple-choice questions in Section A, and ten (10) short answer and coding questions in Section B. All questions in both sections should be attempted.
- **For the multiple choice questions** circle the most correct answer to each question in this exam booklet. Clearly circle only one answer for each question. If you make a mistake, cross it out and circle the correct answer.
- The examination is out of 100 marks, which contribute 40% towards your final assessment. Marks for individual sections and questions are clearly indicated throughout the exam paper.
- For the short answer and coding questions in Section B, you may answer the questions in any order. Boldly number your answers to all questions. Commence all sections on a new page.
- Write your answers to questions from Section B on the lined pages of the exam script. You may use the blank sides of the page for workings, however these pages will not be marked.
- State any assumptions that you make regarding any question.
- The examination end is marked with "END OF EXAMINATION"

Section A: Multiple Choice Questions (40 marks total, 4 marks each)

NOTE: For the purpose of the practice exam I am not writing these as multiple choice because I want to prompt you to ensure you really understand the concepts rather than some bare minimum. These questions are of equivalent difficulty and cover the topics of Section A of the real exam.

Question 1. Compare and relate the von Neumann architecture, Turing machines and Lambda Calculus. What are they? What programming paradigms have they influenced?

Question 2. Consider the following assembly language code:

```
mov eax, 9
push eax
push 7
mov aVariable, 12
push 12
pop anotherVariable
pop eax
```

Immediately after the last line is executed, what value is pointed to by esp and what values are stored in the registers and the variables?

Question 3. In C++, compare and relate RAII and reference counting, and how do these schemes work differently to garbage collection in languages that support it?

Question 4. In lectures we constructed lists of closures in JavaScript (which we called "cons lists").

```
const cons = x=> y=> f=> f(x)(y)
```

What exactly are the symbols cons, x, y, and f in the above code? Use words like function, parameter, argument, closure, apply/applied/application and so on.

Question 5. What do the following terms mean and how do they relate to one another?

referential transparency

pure function

declarative programming

mutable variables

side effects

How can consideration of these ideas lead to more maintainable code?

Question 6. In Haskell what are the names and type definitions of the MINIMAL functions that must be specified for instances of each of the following type classes:

Functor, Applicative, Foldable, Traversable?

Question 7. What is the sequence of mathematical operations that are performed because of each of the following, and what is the final result of each?

i) `foldl (*) 1000 [10, 5, 1]`

ii) `foldr (*) 1000 [10, 5, 1]`

iii) `foldl (/) 1000 [10, 5, 1]`

iv) `foldr (/) 1000 [10, 5, 1]`

Question 8. What type is required for the function that needs to be placed at `_hole` in the following expression? Also give a definition for the expression using <$> and <*>.

```
sequence :: Applicative f => [f a] -> f [a]
sequence = foldr (_hole (:)) (pure [])
```

Question 9. Give examples and definitions for the following terms relating to lambda calculus:

Alpha equivalence

Beta normal form

Beta reduction

Eta conversion

Combinator

Question 10. In MiniZinc, use a forall expression to all the rows of an n by n matrix to sum to the same value. Also declare that the numbers in each row are all different – but to make it more interesting don't use the built-in all different constraint.

Section B: Short Answer and Coding Questions (60 marks total)

NOTE: Once again, the following short answer questions sometimes have several parts in order to cover topics in the exam without giving the actual question directly. Questions worth 4 marks on the real exam have only one part.

Question 11. (4 marks) The Math.pow function in javascript takes two arguments and raises the first to the power of the second, e.g.:

```
Math.pow(3,2)
> 9
```

Write a curried function that uses Math.pow, first with the function keyword, and then give the function again using arrow syntax. Include typescript type annotations. Show how you would use it to square the values in an array.

Question 12. (6 marks) Write a function that can take a function like the one you just wrote for Math.pow, and give back a binary version. Include type annotations.

Question 13. (8 marks total) This question is about **tail recursion** and is in three parts. All code must be self contained, do not use any functions from the Prelude or other libraries.

(i) **(2 marks)** Write a **non-tail recursive** Haskell function to compute the sum of a list of values with type:

```
sum :: [a] -> Integer
```

(ii) **(3 marks)** Now write the function again **using tail recursion**. It should still expose the same type definition:

```
sum :: [a] -> Integer
```

(iii) **(3 marks)** What is the benefit of tail recursion?

Question 14. (6 marks) Given the following Lambda Calculus expressions:

ONE = $\lambda f.\lambda x.(f\ x)$
TWO = $\lambda f.\lambda x.(f\ (f\ x))$
NEXT = $\lambda n.\lambda f.\lambda x.(f\ ((n\ f)\ x))$

Show how the Lambda Calculus can be used to evaluate the math statement:

TWO = NEXT ONE

Give all of the Beta reduction steps in your evaluation using the notation for substitution described in the lectures, i.e. [<variable> := <substitute expression>].

Question 15. (4 marks) Consider the following JavaScript definitions:

```
const cons = head => rest => f => f(head, rest),
      head = list => list((head, rest)=> head),
      rest = list => list((head, rest)=> rest);

const aList = cons(1)(cons(2)(cons(3)(null)))

const forEach = f => l => {
    if (l) {
        f(head(l));
        forEach(f)(rest(l))
    }
}
```

implement the filter function such that the following code produces the output as indicated:

```
forEach(console.log)(filter(a=>a%2==1)(aList))
>1
>3
```

Question 16. (6 marks total) Consider the following TypeScript definitions:

```
interface LazySequence<T> {
  value: T;
  next(): LazySequence<T>;
}

function makeSequence<T>(getNext: (_:T)=>T, initial:T) {
  return function _next(v:T): LazySequence<T> {
    return {
      value: v,
      next: ()=>_next(getNext(v))
    }
  }(initial)
}
```

(i) What parameters would you give to makeSequence to create function which generates the following sequence:

1, -1, 2, -2, 3, -3…

(3 marks).

(ii) What parameters would you give to makeSequence to create function which generates the following sequence of arrays:

[1,2,3], [2,3,4], [3,4,5],…

(iii) **(3 marks)**

Question 17. (6 marks) Write the following Haskell functions in point-free form. On separate lines, show (and name) all of the reduction and conversion steps required to achieve the point-free form *(the actual exam has only one, but I thought I would give you extra practice).*

```
fun1 p l = part (<p) l
fun2 a b c = (a * b) + c
fun3 a b = a `div` (g b)
```

Question 18. (6 marks) Sudoku puzzles typically involve completing a 9 by 9 grid of digits (1..9) such that in each row, column and 3 by 3 subsquare each digit appears only once. For example:

	6	8	4		1		7	
				8	5		3	
	2	6	8		9		4	
		7				9		
	5		1		6	3	2	
	4		6	1				
	3		2		7	6	9	

In the MiniZinc code below which models and solves the above puzzle, complete the constraints for columns and sub-squares.

```
start=[|
  0, 0, 0, 0, 0, 0, 0, 0, 0|
  0, 6, 8, 4, 0, 1, 0, 7, 0|
  0, 0, 0, 0, 8, 5, 0, 3, 0|
  0, 2, 6, 8, 0, 9, 0, 4, 0|
  0, 0, 7, 0, 0, 0, 9, 0, 0|
  0, 5, 0, 1, 0, 6, 3, 2, 0|
  0, 4, 0, 6, 1, 0, 0, 0, 0|
  0, 3, 0, 2, 0, 7, 6, 9, 0|
  0, 0, 0, 0, 0, 0, 0, 0, 0|]

int: S = 3;
int: N = S * S;

set of int: PuzzleRange = 1..N;
set of int: SubSquareRange = 1..S;

array[1..N,1..N] of 0..N: start; %% initial board 0 = empty
array[1..N,1..N] of var PuzzleRange: puzzle;

% fill initial board
constraint forall(i,j in PuzzleRange)(
  if start[i,j] > 0 then puzzle[i,j] = start[i,j] else true endif );

% All different in rows
constraint forall (i in PuzzleRange) (
```

```
  alldifferent( [ puzzle[i,j] | j in PuzzleRange ]) );

% All different in columns.
…

% All different in sub-squares:
…

solve satisfy;
```

Question 19. (6 marks) In Haskell, describe (in words) how Monads can be used for implementing code that conditionally continues computation, generating lists and controlling IO effects. What role does the bind function and do notation play in this regard?

Question 20. (8 marks total)

(i) **(4 marks)** A Rose-tree can store a value, and a list of sub-trees in its nodes – or it can be Nil. Give a generic algebraic data type for such a rose tree that also includes terms in its internal nodes for the minimum and maximum values in any of its subtrees.

(ii) **(4 marks)** Consider a binary search tree with ordered keys and a value associated with each key. Describe in words and diagrams how a pure function can change the value (not the key) stored at one of the leaves in better than linear time and memory.

END OF EXAMINATION