

**THE AUSTRALIAN NATIONAL UNIVERSITY**

*First Semester 2008*

**COMP1100  
(Introduction Programming and Algorithms)**

*Writing Period: 3 hours duration*

*Study Period: 15 minutes duration*

*Permitted Materials: No restrictions, other than excluded electronic devices.*

*Answer ALL questions. Do not spend too much time on any one question.*

*Total marks: 100*

*The questions are followed by labelled blank spaces into which your answers are to be written.*

*Additional answer panels are provided (at the end of the paper) should you wish to use more space for an answer than is provided in the associated labelled panels. If you use an additional panel, be sure to indicate clearly the question and part to which it is linked.*

Name (family name first):

Student Number:

*The following are for use by the examiners.*

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total

## QUESTION 1 [6 marks]

Are each of the following valid Haskell expressions? If so, what are their values? If not, why are they invalid?

(a) `[0] ++ 1 : 2 : 3 : [4,5]`

QUESTION 1(a)

[2 marks]

(b) `zip [[5,4,3],2,1] "Underwood"`

QUESTION 1(b)

[2 marks]

(c) `map last ["Jean", "Luc", "Ponty"]`

QUESTION 1(c)

[2 marks]

## QUESTION 2 [6 marks]

The following functions have been disguised by giving them nonsensical names. Describe the effect and purpose of each function and suggest more meaningful names. Do not give a line by line description of the code.

(a) 

```
moon :: (a -> Bool) -> [a] -> [a]
moon p []           = []
moon p (x:xs)
  | p x             = (x:xs)
  | otherwise      = moon p xs
```

QUESTION 2(a)

[3 marks]

(b) 

```
frank :: Num a => [a] -> Bool
frank []           = True
frank [x]          = True
frank (x:xs)
  | y == x+1       = frank xs
  | otherwise      = False
  where (y:ys) = xs
```

QUESTION 2(b)

[3 marks]

### QUESTION 3 [14 marks]

An obvious way to represent a person's height and weight is with the Haskell type `Float`.

```
type Height = Float
type Weight = Float
```

The **Body Mass Index** (BMI) is a measure of a person's weight, scaled by their height. It is calculated by the formula

$$bmi = \frac{weight}{height^2}$$

(a) Define a Haskell function

```
bmi :: Height -> Weight -> Float
```

to calculate the BMI for a given height and weight using the formula above.

QUESTION 3(a)

[4 marks]

(b) A person has a healthy body mass index if it is between 18.5 and 25.0. Define a Haskell function

```
healthy :: Height -> Weight -> Bool
```

to test whether a person of a given height and weight has a healthy BMI.

QUESTION 3(b)

[5 marks]

- (c) A database of information about a set of people's heights and weights might be represented in Haskell with a list of triples

```
type PeopleStats = [(String, Height, Weight)]
```

where the `String` is the name of the person with the accompanying statistics. Define a Haskell function

```
healthyPeople :: PeopleStats -> [String]
```

which returns a list of the names of all the people in the database with a healthy BMI.

QUESTION 3(c)

**[5 marks]**

## QUESTION 4 [22 marks]

- (a) Describe the purpose and effect of the `update` function defined below. Do not give a line by line description of the code.

```
update :: (a -> a) -> Int -> [a] -> [a]
update f n []      = error "Index out of range."
update f n (x:xs)
  | n == 0         = f x : xs
  | otherwise      = x: update f (n-1) xs
```

QUESTION 4(a)

[4 marks]

- (b) The following function definition uses the `update` defined above but it is missing its type signature.

```
put thing (x,y) grid =
  update (update (const thing) x) y grid
```

Give the type of `put` and describe its purpose and effect. Do not give a line by line description of the code.

QUESTION 4(b)

[6 marks]

(c) The following function definition is missing its type signature.

```
blank xs = all (== ' ') xs
```

What is the type of blank?

QUESTION 4(c)	[4 marks]
---------------	-----------

(d) Given the following two type signatures

```
f :: [a] -> Bool
g :: Eq a => [a] -> Bool
```

What does the `Eq a` part of the type of `g` mean? What does it say about the definition of `g`?

QUESTION 4(d)	[4 marks]
---------------	-----------

(e) Invent definitions for both functions `f` and `g` which have the types given above.

QUESTION 4(e)	[4 marks]
---------------	-----------

## QUESTION 5 [14 marks]

Suppose we wish to catalogue all the inhabitants of a medieval, patriarchal Kingdom by their class. These comprise the *King*, the *Peers*, the *Knights* and the *Peasants*. For each we need to record appropriate information:

The King is simply himself. There is nothing more to say. A Peer has a *division*, a *territory* and a *number* in succession, as in “The 7th Earl of Carlisle.” A Knight or a Peasant has a *name*.

- (a) The divisions of the Peerage are *Duke*, *Earl*, *Viscount* and *Baron*. Define a Haskell enumerated type `Division` to represent those divisions. Since we may wish to compare divisions for equality, and to print them out, add a clause to derive the appropriate type classes.

QUESTION 5(a)	[2 marks]
---------------	-----------

- (b) Names and territories may be represented as `Strings`, but our code will be more readable if we define type synonyms

```
type Name      = String
type Territory = String
```

Based on the description of the inhabitants given above, and using `Division`, `Name` and `Territory`, define a Haskell algebraic type `Person` to represent and record all the appropriate information of all the different kinds of inhabitants of the Kingdom.

QUESTION 5(b)	[4 marks]
---------------	-----------

- (c) In this hierarchical society, some people are considered superior to others. A King is superior to all other classes. Peers are superior to knights and peasants, and knights are superior to peasants. Define a function

```
superior :: Person -> Person -> Bool
```

which implements this hierarchy.

QUESTION 5(c)

[4 marks]

- (d) Define a function

```
sirs :: [Person] -> [Name]
```

which returns the names of all the knights from a list of inhabitants.

QUESTION 5(d)

[4 marks]

## QUESTION 6 [10 marks]

(a) The repeat function creates an infinite list of some value

```
repeat c = c : repeat c
```

The index function returns the nth element of a list

```
index (x:xs) n
  | n == 0      = x
  | otherwise = index xs (n-1)
```

Use these definitions and the principle of induction to prove that for any  $c$  and for all natural numbers  $n$

```
index (repeat c) n = c
```

You first need to prove the base case:

QUESTION 6(a)

[2 marks]

Now state what needs to be proved in the step case:

QUESTION 6(a)	<b>[1 mark]</b>
---------------	-----------------

Lastly, prove the step case:

QUESTION 6(a)	<b>[2 marks]</b>
---------------	------------------

- (b) As an alternative to using the `length` function, we can calculate how many elements are in a list by converting each element to the value 1, then summing the resultant list. We can convert all the elements in a list to 1s by mapping the constant function (`const 1`) over the list.

```
length [] = 0
length (x:xs) = 1 + length xs
```

```
map f [] = []
map f (x:xs) = f x : map f xs
```

```
const a b = a
```

```
sum [] = 0
sum (x:xs) = x + sum xs
```

Use these definitions and the principle of induction to show that both approaches give the same answer. That is, prove that for all lists `ys`,

$$\text{length } ys = \text{sum } (\text{map } (\text{const } 1) \text{ } ys)$$

You first need to state and prove the base case:

QUESTION 6(b)

[2 marks]

Now state what needs to be proved in the step case:

QUESTION 6(b)	<b>[1 mark]</b>
---------------	-----------------

Lastly, prove the step case:

QUESTION 6(b)	<b>[2 marks]</b>
---------------	------------------

## QUESTION 7 [28 marks]

Suppose we are writing some Java classes as part of a system to manage the scheduling and payment of casual workers. The days of the week and the times of day that they work varies, and different people have different rates of pay, presumably based on their age, experience or ability. A good way to represent the days of the week is as an enumeration

```
enum Day {MON, TUE, WED, THU, FRI, SAT, SUN}
```

To keep things simple, we will assume that workers begin and end on the hour, so starting and finishing times can just be given as `Integer` values between 0 and 24. A Java class to represent a single work period could look like:

```
class WorkPeriod {
    Day day;
    Integer start;
    Integer finish;
    ...
}
```

(a) Define a constructor for the `WorkPeriod` class

```
WorkPeriod(Day workDay, Integer from, Integer to)
```

where `workDay` is the day of the work period and `from` and `to` are the start and finish times respectively.

QUESTION 7(a)	[5 marks]

(b) Define a method for the `WorkPeriod` class

```
Boolean weekend()
```

to test whether the day of this work period is on the weekend (SAT or SUN).

QUESTION 7(b)

[5 marks]

(c) Different workers are paid at different rates, and all workers receive time-and-a-half on weekends. Write a method for the `WorkPeriod` class

```
Double pay(Double hourlyRate)
```

The `hourlyRate` argument is the pay rate per hour given to a worker. The `pay` method should calculate the total pay for this work period, at that hourly pay rate. If the work period falls on a weekend (which you should check with the `weekend` method) the total pay should be 1.5 times the normal pay.

QUESTION 7(c)

[5 marks]

- (d) A worker's weekly work schedule could be represented as a list of work periods, as in the following Java class

```
class Schedule {
    ArrayList<WorkPeriod> roster;

    Schedule() {
        roster = new ArrayList<WorkPeriod>();
    }
    void addPeriod(WorkPeriod activity) {
        roster.add(activity);
    }
    ....
}
```

Write a method for the Schedule class

```
Double grossPay(Double hourlyRate)
```

which, using the pay method from the WorkPeriod class, calculates the total pay that a worker will receive for their weekly schedule.

QUESTION 7(d)

[5 marks]

- (e) In object-oriented programming languages such as Java, two of the most fundamental concepts are *classes* and *objects*. How are the concepts of *class* and *object* useful in designing programs?

QUESTION 7(e)

[5 marks]

- (f) In an object-oriented language like Java, what is a *method*?

QUESTION 7(f)

[3 marks]

QUESTION \_\_ ( \_\_ )

QUESTION \_\_ ( \_\_ )

QUESTION \_\_ ( \_\_ )

QUESTION \_\_ ( \_\_ )