Enrolment Number:

The University of Melbourne
School of Computing and Information Systems

Semester 2, 2017 Sample Assessment

COMP30020/COMP90048
Declarative Programming

Reading Time: 15 minutes

Writing Time: 2 hours

Total marks for this paper: 100

This paper has 12 pages.

Examiners' use:

1	2	3	4	5	6	**Total**

Use this page for scratch work. **Work written here will not be assessed.**

Question 1   [12 marks]

For each of the following Haskell expressions, give its **type** (which may be a function type, may include type variables, and may include type class constraints) or indicate that it represents a type error. You need not write anything other than the type, or that it is an error.

(a) (<)

(b) map (+3)

(c) foldr

Use this page for scratch work. **Work written here will not be assessed.**

(d) `Nothing`

| |
| |
| |
| |

(e) `zip [True,True,False]`

| |
| |
| |
| |

(f) `flip filter "hello"`

| |
| |
| |
| |

Use this page for scratch work. **Work written here will not be assessed.**

Question 2   [12 marks]

For each of the following Haskell expressions, give its **value**, or explain why
it will produce an error or fail to terminate. Assume the `Data.List` library
is loaded, which defines the `sort` function.

(a) `map (length < 3) [[1],[1,2,3]]`

(b) `filter (not.(==3)) [1,2,3]`

(c) `let e = head [] in 3`

Use this page for scratch work. **Work written here will not be assessed.**

(d) `map fst $ filter snd $ zip "abcde" [True,True,False,True]`

(e) `head $ sort $ zip [3,0,0,2,0] $ reverse [9,0,0,4,8]`

(f) `map snd $ sort $ zip "decl" [1..]`

Use this page for scratch work. **Work written here will not be assessed.**

Question 3 [30 marks]

Consider the following Haskell type for ternary trees:

```
data Ttree t = Nil | Node3 t (Ttree t) (Ttree t) (Ttree t)
```

Suppose we have a `Ttree` of Doubles and we want a function to find the average of the numbers in the tree. Write a Haskell function which performs this task. If the `Ttree` is empty, your function should return 0.0. Include type declarations for all your functions. To obtain maximum marks, your code should use a single traversal over the tree and have $O(N)$ worst case time complexity.

Use this page for scratch work. **Work written here will not be assessed.**

Question 4   [10 marks]

Give the formal semantics (meaning) of the following Prolog program. Recall that the formal semantics of a logic program is the set of *ground unit clauses* that would give the same answers to all queries as the program itself. English descriptions of the meanings of the programs will receive no credit.

```
p(a).    p(b).

q(X,Y) :- p(X), p(Y).

r(a,c).    r(d,b).

s(X,Y) :- q(X,Y), r(X,_), r(_,Y).
```

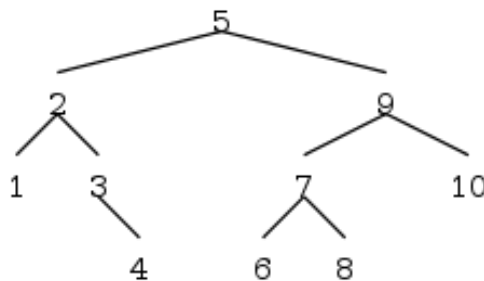| |
|--|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

Use this page for scratch work. **Work written here will not be assessed.**

Question 5   [20 marks]

For this question, we will represent a set of integers as a binary tree in Prolog, using the atom `empty` to represent an empty tree or node, and `tree(L,N,R)` to represent a node with label `N` (an integer), and left and right subtrees `L` and `R`. Naturally, we also insist that `N` be strictly larger than any label in `L` and strictly smaller than any in `R`. We do not require that the tree be balanced. For example,

```
tree(tree(tree(empty, 1, empty),
          2,
          tree(empty, 3, tree(empty, 4, empty))),
     5,
     tree(tree(tree(empty,6,empty),
               7,
               tree(empty,8,empty)),
          9,
          tree(empty, 10, empty)))
```

is one possible representation of the set of numbers from 1 to 10. It might be visualized as

Write a predicate `intset_insert(N, Set0, Set)` such that `Set` is the same as `Set0`, except that `N` is a member of `Set`, but may or may not be a member of `Set0`. That is, either `N` is a member of `Set0` and `Set = Set0`, or `N` is not a member of `Set0` and is a member of `Set`, and other than that, `Set` is the same as `Set0`. This predicate must work as long as `N` is bound to an integer and `Set0` is ground.

Hint: Prolog's arithmetic comparison operators are `<`, `>`, `=<` (not `<=`), and `>=`. You can also use `=` and `\=` for equality and disequality.

Please write your answer on page 9.

Use this page for scratch work. **Work written here will not be assessed.**

Answer to Question 5

Use this page for scratch work. **Work written here will not be assessed.**

Question 6   [16 marks]

Following is a definition of a Prolog predicate to compute the sum of a list of numbers.

```
sumlist([], 0).
sumlist([N|Ns], Sum) :-
        sumlist(Ns, Sum0),
        Sum is N + Sum0.
```

Fill in the blanks in the following transformation of this code to be tail recursive.

```
sumlist(List, Sum) :- [                              ]

sumlist([                            ]).
sumlist([N|Ns], Sum0, Sum) :-
        [                          ],
        sumlist([                        ]).
```

Use this page for scratch work. **Work written here will not be assessed.**

Overflow Answer Page 1

You may use this space to continue any answer, but if you do, indicate *clearly* in your previous answer that you have continued onto this page, or this part of your answer may be overlooked.

Use this page for scratch work. **Work written here will not be assessed.**

Overflow Answer Page 2

You may use this space to continue any answer, but if you do, indicate *clearly* in your previous answer that you have continued onto this page, or this part of your answer may be overlooked.

— **End of Exam** —