final

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2018

BEng Honours Degree in Mathematics and Computer Science Part I
MEng Honours Degree in Mathematics and Computer Science Part I
BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C141=MC141

REASONING ABOUT PROGRAMS

Wednesday 2nd May 2018, 10:00
Duration: 80 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

1 This question is about proof by induction.

a This part is about induction over the definition of lists.

Consider the functions `dup` and `dup2`, defined as follows:

```
dup :: [a] -> [a]
dup xs = dup2 xs []

dup2 :: [a] -> [a] -> [a]
dup2 [] ys = ys
dup2 x:xs ys = dup2 xs (ys++[x])
```

Using induction over the structure of `xs`, prove that

$$(*)\ \forall \texttt{xs:[a]}.\texttt{dup xs} = \texttt{xs}$$

You may need to find, and prove, an auxiliary lemma. In your proofs, state what is given, what is to be shown, and justify all steps. You may use, without proving, some of the the following Lemmas:

(A) $\forall \texttt{z:a, us:[a], vs:[a].}\ \texttt{us++}(\texttt{z:vs}) = (\texttt{us++[z]})\texttt{++vs}$
(B) $\forall \texttt{zs:[a].}\ \texttt{[]++zs} = \texttt{zs} = \texttt{zs++[]}$
(C) $\forall \texttt{us:[a], vs:[a], zs:[a].}\ \texttt{us++}(\texttt{vs++zs}) = (\texttt{us++vs})\texttt{++zs}$

b This part is about induction over user-defined Haskell data types.

Consider the type `T1`, `T2` and the polymorphic type `T3`, defined below:

```
data T1 = C1 Int Char | C2 | C3 [Int] T1
data T2 = C4 T1 | C5 T2 T2 T2
data T3 a b = C6 a b | C7 (T3 a b) (T3 a b)
                     | C8 [a] (T3 a b)
```

Assuming predicates $P \subseteq \texttt{T1}$, and $Q \subseteq \texttt{T2}$, and $R \subseteq (\texttt{T3 Bool Bool})$, give induction principles which prove that

i) $\forall \texttt{t1} : \texttt{T1}.\ P(\texttt{t1})$

ii) $\forall \texttt{t2} : \texttt{T2}.\ Q(\texttt{t2})$

iii) $\forall \texttt{t3} : (\texttt{T3 Bool Bool}).\ R(\texttt{t3})$

c This part is about induction over data types and over the inductive definition of relations.

Consider the following definition of the type `T`:

```
data T   = Nd [T]  | Lf
```

Assume a predicate $P \subseteq \mathtt{T}$, and another predicate $Q \subseteq \mathtt{T} \times \mathtt{T}$.

i) Give an inductive principle that proves that

$$\forall \mathtt{t} : \mathtt{T}.\ P(\mathtt{t}).$$

You might like to use an auxiliary predicate $All \subseteq [\mathtt{T}]$, defined as follows:

$All(\mathtt{ts})$ iff $\forall \mathtt{ts}', \mathtt{ts}'' : [\mathtt{T}].\forall \mathtt{t} : \mathtt{T}.[\ \mathtt{ts} = \mathtt{ts}'{+}{+}[\mathtt{t}]{+}{+}\mathtt{ts}'' \ \rightarrow\ P(\mathtt{t})\ ]$.

It is possible, however, to approach this question without such a predicate.

ii) Assume a predicate $R \subseteq \mathtt{T} \times \mathtt{T}$, defined as follows:

A) $\forall \mathtt{t}, \mathtt{t}', \mathtt{t}'' : \mathtt{T}.$
$[\ R(\mathtt{t}, \mathtt{t}') \wedge R(\mathtt{t}', \mathtt{t}'') \ \rightarrow\ R(\ \mathtt{t}, \mathtt{t}''\ )\ ]$

B) $\forall \mathtt{t}, \mathtt{t}' : \mathtt{T}.\forall \mathtt{ts} : [\mathtt{T}].$
$[\ R(\mathtt{t}, \mathtt{t}') \ \rightarrow\ R(\ \mathtt{Nd}\,(\mathtt{t} : \mathtt{ts}), \mathtt{Nd}\,(\mathtt{t}' : \mathtt{ts})\ )\ ]$

C) $\forall \mathtt{ts} : [\mathtt{T}]$
$[\ R(\ \mathtt{Nd}\,((\mathtt{Nd}\,[]) : \mathtt{ts}), \mathtt{Nd}\,\mathtt{ts}\ )\ ]$

D) $\forall \mathtt{ts}, \mathtt{ts}' : [\mathtt{T}].$
$[\ R(\ \mathtt{Nd}\,((\mathtt{Nd}\,(\mathtt{Lf}{:}\mathtt{ts})){:}\mathtt{ts}'\ ), \mathtt{Nd}\,((\mathtt{Nd}\,\mathtt{ts}){:}\mathtt{Lf}{:}\mathtt{Lf}{:}\mathtt{ts}')\ )\ ]$

Based on the definition of R, give an inductive principle which proves that

$$\forall \mathtt{t}, \mathtt{t}'' : \mathtt{T}.[\ R(\mathtt{t}, \mathtt{t}') \ \rightarrow\ Q(\mathtt{t}, \mathtt{t}')\ ].$$

The three parts carry, respectively, 45%, 25%, and 30% of the marks.

2 This question is about method calls and loops.

Consider the following Java method `shufflesort` written by a student from the Mathematics Department:

```
void shufflesort(int[] a)
// PRE: a ≠ null                                          (P)
// POST: a ∼ a₀ ∧ Sorted(a[0..a.length))                  (Q)
{
    boolean done = false;
    // INV: ???                                           (I₁)
    // VAR: ???                                           (V₁)
    while(!done){
        shuffle(a);
        done = true;
        int i = 0;
        // INV: a ∼ a₀ ∧ 0 ≤ i < a.length                 (I₂)
                    ∧ [done ⟶ Sorted(a[0..i + 1))]
        // VAR: ???                                       (V₂)
        while(i < a.length - 1){
            done = done && (a[i] <= a[i+1]);
            i++;
        }
    }
}
```

This method aims to perform an in-place sort on an array of integers, making use of an auxiliary library method `shuffle` that permutes the elements of an array. The implementation of the `shuffle` method is not known, but it is claimed that it satisfies the following specification:

```
void shuffle(int[] a){
//PRE: a ≠ null
//POST: a ∼ a₀
    ...
}
```

a Unfortunately, the author has not specified the `shufflesort` method well, forgetting to define the *Sorted* predicate used in the postcondition.
Provide a definition of the *Sorted* predicate so that it describes a sorted array slice $\mathtt{a}[x..y)$.

b A Computing student wants to use this function, but first wants to be sure that on termination of the outer while-loop the postcondition will be satisfied.

i) Give a loop invariant I_1 for the outer while-loop that is appropriate to show partial correctness of the `shufflesort` method.
(You do not need to prove anything yet.)

ii) Prove that if the outer while-loop terminates then the post-condition of the `shufflesort` method holds.
State clearly what is given and what you need to show.

c Given the invariant I_2 for the inner while-loop in the `shufflesort` method, prove that the inner while-loop re-establishes this invariant.
State clearly what is given and what you need to show.

d The computing student now wants to check if the `shufflesort` method is guaranteed to terminate. First we consider the inner while-loop.

i) Give a loop variant V_2 for the inner while-loop.

ii) Using your answer to part (i), prove that the inner while-loop terminates.

e There is a problem with the termination of the outer while-loop of the `shufflesort` method. Briefly describe the problem.

f The Computing student reads the documentation for the `shuffle` library method in detail and discovers that there is a version of the method that takes an additional integer parameter:

```
void shuffle(int[] a, int x){
//PRE: a ≠ null ∧ 0 ≤ x < (a.length)!
//POST: Shuff(a₀, a, x)
    ...
}
```

If this version of `shuffle` can be used to guarantee termination of the outer while-loop, what property must be true about the $Shuff(a, b, k)$ predicate?

i) Briefly describe this property in English.

ii) Give a corresponding formal (logical) property.

iii) Modify the `shufflesort` code on lines 5-9 so that termination of the `shufflesort` method is now guaranteed.
(You do not need to modify the invariant I_1 or variant V_1.)

The six parts carry, respectively, 10%, 15%, 25%, 20%, 5%, and 25% of the marks.