IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2019

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BEng Honours Degree in Mathematics and Computer Science Part I
MEng Honours Degree in Mathematics and Computer Science 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 1st May 2019, 10:00
Duration: 80 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

1 This question is about proof by induction.

The following code describes a Haskell data structure `BT` which stands for binary trees whose leaves are `Box`-es, and a function `rev` which reverses such trees.

```
data BT = Box | Nd BT BT

rev :: BT -> BT
rev Box = Box
rev (Nd t1 t2) = Nd (rev t2)  (rev t1)
```

a Write out the result of

```
rev (Nd Box (Nd (Nd Box Box) Box)).
```

(You do not need to show any intermediate steps.)

b Assume some predicate $P \subseteq \texttt{BT}$, and another predicate $Q \subseteq \texttt{BT} \times \texttt{BT}$.

i) Write out the induction principle for `t`:`BT`, which gives

$$\forall \texttt{t} : \texttt{BT}.P(\texttt{t}).$$

ii) Write out the induction principle for `t`:`BT`, which gives

$$\forall \texttt{t} : \texttt{BT}.\forall \texttt{t}' : \texttt{BT}.Q(\texttt{t},\texttt{t}').$$

The predicate $Palin \subseteq \texttt{BT}$ is defined as follows:

$$Palin(\texttt{t}) \triangleq \texttt{t} = \texttt{rev t}.$$

Consider the function `checkPalin` defined as follows:

```
checkPalin :: BT -> Bool
checkPalin Box = True
checkPalin (Nd t1 t2) = check t2 t1

check :: BT -> BT -> Bool
check Box Box = True
check (Nd t1 t2) (Nd t3 t4) = (check t1 t4)
                              &&(check t3 t2)
check t1 t2 = False
```

We want to prove that `checkPalin` establishes whether *Palin* holds, *i.e.,* we want to show property (A) defined below. In order to prove (A), we first formulate a stronger property, (B), also defined below:

(A) $\quad \forall \texttt{t} : \texttt{BT}.[\ \mathit{Palin}(\texttt{t}) \longleftrightarrow \texttt{checkPalin t}\ ]$

(B) $\quad \forall \texttt{t},\texttt{t}' : \texttt{BT}.[\ \texttt{check t t}' \longleftrightarrow \texttt{t} = \texttt{rev t}'\ ]$

c In up to two sentences outline the proof that $(B) \longrightarrow (A)$.

d Prove (B). Write what is to be shown, state which variables are taken arbitrary, and justify each step.

In parts c) and d) above you may, if you want to, use the following facts without further proof:

$F1$ $\quad \forall \texttt{t} : \texttt{BT}.[\ \texttt{rev (rev t)} = \texttt{t}\ ]$

$F2$ $\quad \forall \texttt{t},\texttt{t}' : \texttt{BT}.[\ \texttt{t} = \texttt{rev t}' \longleftrightarrow \texttt{t}' = \texttt{rev t}\ ]$

The function `check2` compares lists of pairs of trees:

```
check2 :: [ (BT,BT) ] -> Bool
check2 [] = True
check2 ((Box,Box):ts) =  check2 ts
check2 (((Nd t1 t2),(Nd t3 t4)):ts) =
                 check2 ((t1,t4):(t2,t3):ts)
check2 ts = False
```

`check2` has the property that

(C) $\quad \forall \texttt{t},\texttt{t}' : \texttt{BT}.[\ \texttt{check2 ((t t'):[])} \longleftrightarrow \texttt{t} = \texttt{rev t}'\ ]$

To prove (C) we generalize to something which has the form

(D) $\quad \forall \texttt{pts} : [(\texttt{BT},\texttt{BT})].[\ \texttt{check2 pts} \longleftrightarrow Q(\texttt{pts})\ ]$

e Define the property $Q(\texttt{pts})$. (You do *not* need to prove anything.)

The five parts carry, respectively, 5%, 10%, 15%, 55%, and 15% of the marks.

2 This question is about loops.

Consider a variant of the Fibonacci sequence fib_N (where $N \geq 2$) such that $fib_N(0) = 0$, $fib_N(1) = 1$ and then each subsequent number is the sum of the previous N numbers (or as many as exist).

We can actually define this sequence over *all* integers ($n \in \mathbb{Z}$) as follows:

$$fib_N(n) \;=\; \begin{cases} -1 & \text{if } n < 1-N \\ 1 & \text{if } n = 1-N \\ 0 & \text{if } 1-N < n < 0 \\ \sum_{j=n-N}^{n-1} fib_N(j) & \text{otherwise} \end{cases}$$

For example when $N = 4$:

n	...	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	...
$fib_4(n)$	...	-1	-1	1	0	0	0	1	1	2	4	8	15	...

The following Java method `fibNacci(n,N)` calculates the $\mathtt{n}^{\text{th}}$ number in the $fib_{\mathtt{N}}$ sequence, using an internal array to avoid needing to make recursive calls:

```
1  public static int fibNacci(int n, int N)
2  // PRE: n ≥ 0 ∧ N ≥ 2                                              (P)
3  // POST: r = fib_N(n)                                              (Q)
4  {
5      int[] a = new int[N];
6      a[1] = 1;
7      int i = 0;
8      // INV: a.length = N ∧ N ≥ 2                                   (J1)
9      //         ∧ 0 ≤ i ≤ n                                        (J2)
10     //         ∧ ∀k ∈ [0..N).[ a[k] = Σ_{j=i-N+k}^{i-1} fib_N(j) ] (J3)
11     // VAR: n − i                                                  (U)
12     while (i < n){
13         int cnt = 0;
14         int val = a[0];
15         // INV: a.length = N ∧ N ≥ 2 ∧ ???                         (I)
16         // VAR: ???                                                (V)
17         while (cnt < N-1){
18             a[cnt] = a[cnt+1] + val;
19             cnt++;
20         }
21         a[cnt] = val;
22         i++;
23     }
24     return a[0];
25 }
```

Notice from the pre-condition that `fibNacci(n,N)` can only be called on natural numbers `n,N` $\in \mathbb{N}$ with `N` ≥ 2.

The code above does not modify the input parameters `n` or `N`, so you do not need to distinguish between `n`, $\texttt{n}_{pre}$ or $\texttt{n}_{old}$ throughout this question (similarly for `N`).

You may also assume that `int[] a = new int[N]` creates a new integer array of length `N`, with all of its contents initially set to `0`, and returns its address to the variable `a`.

a Show that `fibNacci(6,4)` $=$ `15` by writing out the states of the loop counters `i` and `cnt`, the temporary variable `val` and the internal array `a` at the end of each outer loop iteration (line 23).

Give your answer in a table as below (we have completed the first row for you):

i	val	cnt	a[..]
1	0	3	[1,0,0,0]
⋮	⋮	⋮	⋮

b Show that on termination of the outer loop and after executing the code on line 24 the post-condition Q of the `fibNacci` method is established.
State clearly what is given and what you need to show.

c Unfortunately, the author has not fully specified the inner loop of the method.

i) Give a loop variant V for the inner loop that is appropriate to show termination. (You do *not* need to prove anything.)

ii) Complete the loop invariant I for the inner loop so that it is appropriate to show total correctness. (You do *not* need to prove anything.)

[**Hint:** *We suggest you add four or five further conjuncts. The first should bound* `i`*, the second should bound* `cnt`*, the third should define* `val` *and the last one or two should describe the array contents (modified and unmodified parts).*]

d Set up a proof to show that on termination of the inner loop and after executing the code on lines 21 and 22 the invariant J of the outer loop is re-established. State clearly what is given and what you need to show, but do *not* prove anything.

e When $2 \leq n \leq N+1$ the value of $fib_N(n)$ can actually be calculated directly (i.e. without using recursion). Give an expression that describes this behaviour. That is, find some ??? such that $\forall n \in [2..N+1].[\ fib_N(n) = ???\ ]$

[**Hint:** *Try running* `fibNacci(n,N)` *to generate the first few terms of the* fib_N *sequence for a large value of* N *and look for a pattern.*]

The five parts carry, respectively, 15%, 20%, 35%, 20%, and 10% of the marks.