IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2017

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

Friday 12 May 2017, 14:00
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 `rev2` and `rev`, defined as follows:

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

rev2 :: [a] -> [a] -> [a]
rev2 [] ys = ys
rev2 x:xs ys = rev2 xs x:ys
```

Prove that

$(*)$ $\forall$`xs:[a]`.$\forall$`ys:[a].rev2 xs ys` $=$ `(rev xs) ++ ys`,

using induction over the structure of `xs`.

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

(A) $\forall$`z:a, us:[a], vs:[a]. us++(z:vs)` $=$ `(us++[z])++vs`

(B) $\forall$`zs:[a]. []++zs` $=$ `zs` $=$ `zs++[]`

(C) $\forall$`us:[a], vs:[a], zs:[a]. us++(vs++zs)` $=$ `(us++vs)++zs`

b This part is about induction over the definition of functions. Consider the functions `f`, `g`, and `h`, defined below:

```
f :: Int -> Int
f n =
    | 0<=n && n<3 = 10 + 10*n
    | otherwise   = f(n-1)*f(n-3)

g :: Int -> Int
g n =
    | 0<=n && n<3 = 10 + 10*n
    | otherwise   = h(n,2,30,20,10)

h :: (Int,Int,Int,Int,Int) -> Int
h(n,cnt,k1,k2,k3)
    | n==cnt      = k1
    | otherwise   = h(n,cnt+1,k1*k3,k1,k2)
```

We will study various proof steps in establishing that $(A)\ \forall n:\mathbb{N}.\ \mathtt{f}\,n = \mathtt{g}\,n$. In particular:

i) Assume a predicate $P \subseteq \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$, and write out the inductive principle as applied to the definition of h, which implies that:

$$(B) \quad \forall n, cnt, k1, k2, k3, r : \mathbb{Z}.$$
$$[\, \mathtt{h}(n, cnt, k1, k2, k3) = r \;\rightarrow\; P(n, cnt, k1, k2, k3, r) \,]$$

ii) Write out the proof schema that allows us to prove that:

$$(C) \quad \forall n, cnt, k1, k2, k3, r : \mathbb{Z}.$$
$$[\quad \mathtt{h}(n, cnt, k1, k2, k3) = r \;\rightarrow$$
$$[\, n \geq 3 \,\land\, k1 = \mathtt{f}(cnt) \,\land\, k2 = \mathtt{f}(cnt-1) \,\land\, k3 = \mathtt{f}(cnt-2)$$
$$\rightarrow\; r = \mathtt{f}(n) \,]\;]$$

Remember that writing out a proof schema only requires you to state what variables are taken with arbitrary values, what is given, and what is to be shown for the Base Case(s) and the Inductive Steps(s). It does *not* require you to prove anything.

iii) For which values of $n : \mathbb{Z}$ does the term $\mathtt{g}\, n$ terminate?

iv) Briefly outline how we can prove termination for values you have specified in your answer in part iii. (Use one or two sentences.)

v) Briefly describe how we can prove (A). (Use one or two sentences.)

The two parts carry, respectively, 45%, and 55% of the marks.

2 This question is about method calls and loops.

Consider the following Java method `factors` written by a student from UCL:

```
int[] factors(int n)
// PRE: ???                                          (P)
// POST: ∀y ∈ [0..r.length). ∃m ∈ ℕ. m ∗ r[y] = n    (Q)
{
    int[] fs = new int[n/2];
    int pos = 0;
    int curr = n;
    int cand = 2;
    // INV: ??? ∧ ??? ∧ ??? ∧ ???                    (I)
    // VAR: ???                                      (V)
    while(curr > 1){
        int val = (curr % cand);
        if(val == 0){
            fs[pos] = cand;
            curr = curr/cand;
            pos++;
        } else {
            cand++;
        }
    }
    return fs;
}
```

which returns an array containing the factors of the input `n` and where `%` and `/` are the usual Java modulus and integer division operators, respectively.

a Write the arrays returned after running the following method calls:

i) `factors(1)` iii) `factors(7)`

ii) `factors(4)` iv) `factors(10)`

b Unfortunately, the author has not specified the method well, forgetting to give it a precondition and providing an invalid postcondition.

i) Give an example integer input `i` where running `factors(i)` would throw an exception.

ii) Briefly describe why this exception would be generated.

iii) Give a precondition P for `factors` that would rule out just those inputs that would generate this exception.

iv) Briefly describe the problem with the postcondition Q for `factors`.

v) Give an improved postcondition for `factors` that captures the intended behaviour of the method **and** is satisfied by the above implementation.

c An Imperial Computing student wants to use this function, so writes the following wrapper to correct the code and provides a new postcondition:

```
int[] wrapper(int n)
// POST: $\Pi_{k=0}^{\mathbf{r}.\texttt{length}-1}\mathbf{r}[k] = \texttt{n}$          $(R)$
{
    int[] a = factors(n);
    int len = search(a,0);
    int[] ret = new int[len];
    int cnt = 0;
    while(cnt < ret.length){
        ret[cnt] = a[cnt];
        cnt++;
    }
    return ret;
}
```

where `search(a,x)` returns the array index of the first occurrence of `x` in `a`, or `a.length` if `x` does not occur in `a`.

The original author complains that the wrapper completely changes the intended behaviour of the code. Prove that they are incorrect (i.e. that R implies Q). State clearly what is given, what you need to show and justify each step.

d To be sure that the `wrapper` method is totally correct, the Imperial Computing student wants to be sure that the while-loop in `factors` is totally correct.

i) Complete the loop invariant I so that it is appropriate to show partial correctness of the `factors` method with respect to the midcondition $\Pi_{k=0}^{\texttt{pos}-1}\texttt{fs}[k] = \texttt{n}$ added at line 20. (You do not need to prove anything.)

[**Hint:** *There are four conjuncts. The first three should describe important variables in the code and the last should describe the array* `fs`*.*]

ii) Write a loop variant V that is appropriate to show total correctness of the `factors` method. (You do not need to prove anything.)

e An Imperial JMC student reads both of the methods above and remarks that *"All elements of the array returned by* `wrapper` *are actually prime"*.

i) Propose a modification to the postcondition R for `wrapper` that would describe this additional property.

ii) Briefly justify why the JMC student is correct. (You do not need to prove anything.)

The five parts carry, respectively, 10%, 25%, 25%, 25%, and 15% of the marks.