IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2016

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

Thursday 5 May 2016, 10:00
Duration: 80 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

1 This question is concerned with proof by induction, derivation of inductive principles, and the formulation of useful auxiliary lemmas.

a Consider the functions `rev` and `flip` defined as follows:

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

flip :: [Int] -> [Int]
flip [] = []
flip (i:is) = (-i):(flip is)
```

Prove that

$\forall$ `is:[Int]. rev(flip is) = flip(rev is).`

by structural induction on `is`.

In the proof, state what is given, what is to be shown, what is assumed, which variables are taken arbitrarily, and justify each step. You may use the facts that:

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

(B): $\forall$ `is,js:[Int]. flip(is++js) = (flip is)++(flip js)`

(C): $\forall$ `i:Int. flip[i]=[-i]`

b Consider the datatype `DT` defined as follows:

```
data DT = C1 Int | C2 DT [Char] | C3 DT [DT]
```

Write the inductive principle which implies that $\forall$ `dt:DT.`P(`dt`), for any property $P \subseteq$ `DT`.

c Now consider the partial function $F : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined as follows:

$$\mathtt{n{=}cnt} \rightarrow \quad F(\mathtt{n,cnt,acc}) = \mathtt{acc}$$
$$\mathtt{n{\neq}cnt} \rightarrow \quad F(\mathtt{n,cnt,acc}) = F(\mathtt{n,cnt+1,(cnt+1)*acc})$$

We want to prove that

$$(*) \quad \forall \mathtt{n,p} \in \mathbb{N}.\, [\, F(\mathtt{n,1,1}) = \mathtt{p} \rightarrow \mathtt{p} = \prod_{i=1}^{\mathtt{n}} i \,],$$

where $\prod_{i=j}^{m} i$ is the product of the numbers between j and m. In order to prove $(*)$ we need to prove a stronger assertion, $(**)$, which implies $(*)$, of the form

$$(**) \quad \forall \mathtt{n,p,cnt,acc} \in \mathbb{N}.\, [\, F(\mathtt{n,cnt,acc}) = \mathtt{p} \ \rightarrow\ Q \,],$$

where Q is some assertion containing `p`.

Write out the assertion Q. Do not prove anything.

The three parts carry, respectively, 65%, 15%, and 20% of the marks.

This page intentionally left blank

2 This question is about method calls and loops.

Consider the following Java method `arrayRng`:

```
1  int arrayRng(int[] a)
2  // PRE: a ≠ null ∧ a.length > 0             (P1)
3  // POST: a ≈ a0 ∧ r = max(a) − min(a)       (Q1)
4  {
5      int s = minloc(a);
6      // MID: ???                             (M1)
7      int b = maxloc(a);
8      // MID: ???                             (M2)
9      return a[b] - a[s];
10 }
```

where:

$$min(\texttt{a}) = min\{z \mid \exists k \in \mathbb{N}.\, 0 \le k < \texttt{a.length} \land \texttt{a}[k] = z\}$$

$$max(\texttt{a}) = max\{z \mid \exists k \in \mathbb{N}.\, 0 \le k < \texttt{a.length} \land \texttt{a}[k] = z\}$$

The method makes use of two auxiliary methods `minloc` and `maxloc`. Given an array of integers `n`, these methods return the location of the minimum and maximum elements of that array, respectively. The implementations of `minloc` and `maxloc` are not known, but they have been proven to satisfy the following specifications:

```
int minloc(int[] n)
// PRE: n ≠ null ∧ n.length > 0                 (P2)
// POST: n ≈ n0 ∧ n[r] ≤ n[0..n.length)         (Q2)

int maxloc(int[] n)
// PRE: n ≠ null ∧ n.length > 0                 (P3)
// POST: n ≈ n0 ∧ n[r] ≥ n[0..n.length)         (Q3)
```

a Given an input array $\texttt{b} = [1, -4, 5, 1, 8]$ write the values returned after running the following method calls:

i) `minloc(b)`

ii) `maxloc(b)`

iii) `arrayRng(b)`

b Write mid-conditions M_1 and M_2 that hold at lines 6 and 8 of `arrayRng` respectively and are appropriate to show partial correctness of `arrayRng`. (You do not need to prove anything.)

c Prove that your mid-condition M_1 from part (b) is established after the call to `minloc(a)` on line 5. State clearly what is given and what you need to show.

d A programmer decides that it is inefficient to traverse the array twice with the helper methods and so proposes the use of the following in-line loop to replace the code on lines 5 - 7 in `arrayRng`:

```
int s = 0;
int b = 0;
int cnt = 1;
// INV: ??? ∧ ??? ∧ ??? ∧ ???                  (I)
// VAR: ???                                    (V)
while(cnt < a.length){
    if(a[cnt] < a[s]){ s = cnt; }
    if(a[cnt] > a[b]){ b = cnt; }
    cnt++;
}
```

i) Complete the loop invariant I so that it is appropriate to show partial correctness of the modified `arrayRng` method.
(You do not need to prove anything.)

[**Hint:** *There are four conjuncts. The first should bound* `cnt`*, the second should describe the state of the array* `a`*, and the third and fourth should describe useful properties about* `s` *and* `b`*, respectively.*]

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

e Another programmer wants to strengthen the post-condition Q_1 of `arrayRng` to:

$$Q_1' \quad \equiv \quad \mathtt{a} \approx \mathtt{a}_0 \ \land \ \mathbf{r} = max(\mathtt{a}) - min(\mathtt{a}) \ \land \ \mathbf{r} \leq max(\mathtt{a})$$

This is **not** currently true.

i) Propose a modification to the method's pre-condition P_1 that will guarantee partial correctness of the method with the new post-condition Q_1' given above.

ii) Briefly justify this modification.
(You do not need to prove anything.)

The five parts carry, respectively, 15%, 15%, 30%, 30%, and 10% of the marks.