COL 100	Minor 1	**Feb 05, 2020**
Name ____________	*EntryNo* ____________	*Group* ____________

Note (i) Write your answers in the blank parts of the question sheet including the back of the sheet. Do all rough work in separately provided paper. (ii) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.

Problem 1: 20 marks

Suppose your favourite programming language can only perform an add (`add`: $\mathbb{N}\times\mathbb{N} \rightarrow \mathbb{N}$) operation, but not multiply.

1. Then, define multiplication (`mul`: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$) in terms of the `add`: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ function (`add(a,b)` returns $a + b$).

   Solution: Using the invariant: $(0 \leq i \leq a) \wedge (sum = i * b)$

```
fun mult(a,b) =
        if (b = 0) then 0
        else
          let
             fun mult_iter(i,a,b,sum) = if (i = a) then sum
                                  else mult_iter(add(i,1), a, b,add(sum,b))
          in
              mult_iter(0,a,b,0)
          end;
```

   Alternatively,

```
fun mult(a,b) =
        if (b = 0) then 0
        else
          let
               fun mult_rec(i,a,b) =
                          if (i = a) then 0
                          else add(mult_rec(add(i,1), a, b), b)
          in
             mult_rec(0,a,b)
          end;
```

2. Derive the time and space complexity of this algorithm in terms of number of invocations of the recursive function in the definition of `mult`?

   Solution: Number of calls to `mult_iter` follows the recurrence relation $T(a) = T(a-1) + 1$, with $T(0) = 1$. Thus, the total number of recursive calls is a. Space complexity is $O(1)$ for the iterative version and $O(b)$ for the recursive version.

3. Suppose we include, in addition to `add`, functions `dec` : $\mathbb{P} \rightarrow \mathbb{N}$, which decrements an integer by 1, `even` : $\mathbb{N} \rightarrow \mathbb{B}$, which tests whether or not a given number is even, `double` : $\mathbb{N} \rightarrow \mathbb{N}$, which doubles an integer, and `halve` : $\mathbb{N} \rightarrow \mathbb{N}$, which divides an (even) integer by 2. Can you construct a more efficient algorithm for `mult`.

   Solution:

   ```
    fun mult(a,b) =
            if (b = 0) then 0
            else if even(b) then
                   double(mult(a,halve(b)))
            else   add(a,mult(a,dec(b)));
   ```

   Time complexity is $O(\log_2 b)$.

Problem 2: 10 marks

Write an `ML` program to count the number of ways for a rook to move from the southwest corner of a $p \times q$ chessboard to the northeast corner by moving one square at a time eastward or northward only. Note that rook is a chess piece that can move horizontally and vertically on a chess board. Give a recursive formulation and do not use a formula.
Solution:

Observation: It is a sum of all solutions that start by taking a step northward and those that start by taking a step eastward. Thus, if the initial problem was posed as $rooks(p,q)$, then the inductive formulation is $rooks(p,q) = rook(p-1,q) + rooks(p,q-1)$. But what are the basis conditions? Note that $rooks(1,1) = 1$. But $rooks(p,q) = 0$ when either $p = 0$ or $q = 0$. Thus, the SML program is as follows:

```
fun rooks(p,q) =
         if p = 0 orelse q = 0 then 0
         else if p =1 andalso q = 1 then 1
              else rooks(p-1, q) + rooks(p, q-1)


fun rook(p,q) =
        if ((p = 1) orelse (q = 1)) then 1
        else rook(p-1,q)+rook(p,q-1);

fun C(m,n) =
        if ((m = n) orelse (n = 0)) then 1
        else C(m-1,n-1)+C(m-1,n);

fun rook1(p,q) = C(p+q-2,p-1);
```

Problem 3: 15 marks

Consider the following algorithm for dividing an integer $x \geq 0$ by an integer $y > 0$ and determining $q \geq 0$ and $0 \leq r < y$ such $x = qy + r$.

```
fun divide(x,y) =
        if (x = 0) then (0,0)
        else  let   val (q,r) = divide(x div 2,y);
                    val q1 = 2*q;
                    val r1 = 2*r
              in
                    if (x mod 2 = 1) then
                           let val r2 = r1+1
                           in
                              if (r2 < y) then (q1,r2)
                              else (q1+1,r2-y)
                           end
                    else if (r1 < y) then (q1,r1)
                         else (q1+1,r1-y)
              end;
```

Prove that the algorithm is correct. Estimate the time complexity of the above algorithm as a function of n, where the number of digits/bits in x is n.
Solution:
TST: `divide`(x, y) for $x \geq 0$ and $y > 0$ returns (q, r) such that $x = qy + r$, $q \geq 0$ and $0 \leq r < y$.

Proof. By **PMI V3** on x.

Basis: For $x = 0$, `divide`(x, y) returns $(q, r) = (0, 0)$.

IH: For $0 \leq z < x$, `divide`(z, y) returns (q, r) such that $z = qy + r$, $q \geq 0$ and $0 \leq r < y$.

IS:

1. Let $x = 2k$. Then, by IH, `divide`$(x \text{ div } 2, y)$ returns (q, r) such that $k = qy + r$, $q \geq 0$ and $0 \leq r < y$. Thus, by algorithm, $x = 2k = q_1y + r_1$, where $q_1 = 2q$ and $r_1 = 2r$. If $r_1 \leq y$ the algorithm returns (q_1, r_1) as required. If $r_1 > y$ then $r_1 - y = 2r - y < y$ because $r < y$. Thus $x = (q_1 + 1)y + (r_1 - y)$ and the algorithm returns $(q_1 + 1, r_1 - y)$ as required.
2. Let $x = 2k + 1$. Then, by IH, `divide`$(x \text{ div } 2, y)$ returns (q, r) such that $k = qy + r$, $q \geq 0$ and $0 \leq r < y$. Thus, by algorithm, $x = 2k + 1 = q_1y + r_1 + 1 = q_1y + r_2$, where $q_1 = 2q$ and $r_2 = 2r + 1$. If $r_2 \leq y$ the algorithm returns (q_1, r_2) as required. If $r_2 > y$ then $r_2 - y = 2r + 1 - y < y$ because $r < y$. Thus $x = (q_1 + 1)y + (r_2 - y)$ and the algorithm returns $(q_1 + 1, r_2 - y)$ as required.

□

The recurrence that describes the run time behaviour of the above algorithm is $T(x) = T(x \text{ div } 2) + O(1)$; $T(0) = 0$, which solves to $T(x) = O(\log_2 x)$.

Problem 4: 15 marks

In second lab assignment we had defined the *integer square root* of an integer n as the integer k such that $k^2 \leq n < (k+1)^2$, and computed it using the following inductive process:

- Compute the integer square root i of $m = n \ div \ 4$ recursively. We then have that $i^2 \leq m < (i+1)^2$
- Since m and i are integers we have that $(m+1) \leq (i+1)^2$. We thus have $(2i)^2 \leq 4m \leq n < 4m + 4 \leq (2i+2)^2$. Hence we have that the integer square root of n is either $2i$ or $2i+1$.

1. Now write an iterative version corresponding to the above algorithm using the following invariant:

   $INV : c_0 = 4^{k_0}$ such that $n \ div \ 4^{k_0} > 0$ and $n \ div \ 4^{k_0+1} = 0$. After k iterations, $0 \leq k \leq k_0$, $c = c_0 \ div \ 4^k$ and $i^2 \leq n \ div \ (c*4) < (i+1)^2$.

   You may use the following helper function:

   ```
   fun maxpow(n) =
         if (n div 4 = 0) then 1
         else 4*maxpow(n div 4);
   ```

   Solution: Let us first figure out how many iterations are required. Clearly it is k such that $n \ div \ 4^k > 0$ and $n \ div \ 4^{k+1} = 0$. This suggests the following invariant:

   An iterative `ML` program based on the above invariant can be given as:

   ```
   fun intsqrt(n) =
   let fun maxpow(n) =
           if (n div 4 = 0) then 1
           else 4*maxpow(n div 4);
      fun iter(i,n,c) =
           if (c = 0) then i
           else if (2*i+1)*(2*i+1) > (n div c) then iter(2*i,n,c div 4)
           else iter(2*i+1,n,c div 4);
   in
      iter(0,n,maxpow(n))
   end;
   ```

2. Derive the number of steps required by the algorithm.

   Solution: The number of steps required (according to the derivation of the invariant) is $k_0 = O(\log_4 n)$.

Problem 5: BONUS: 10 marks

The solution to this problem may be submitted on Moodle by end of today (23:59).

A coffee can contains some black beans and white beans. The following process is to be repeated as long as possible:

- Randomly select two beans from the can. If they are the same colour, throw them out, but put another black bean in (Assume that enough black beans are available to do this). If they are of different colours, place the white one back into the can and throw the black one away.

Execution of this process reduces the number of beans in the can by one. Repetition of this process must terminate with exactly one bean in the can. What can you say about the colour of this last bean depending on the number of black beans and the number of white beans that were there in the can to start with? Give reasons for your answer.