Computer Science 312

Fall 2006

Prelim 2 SOLUTIONS

April 18, 2006

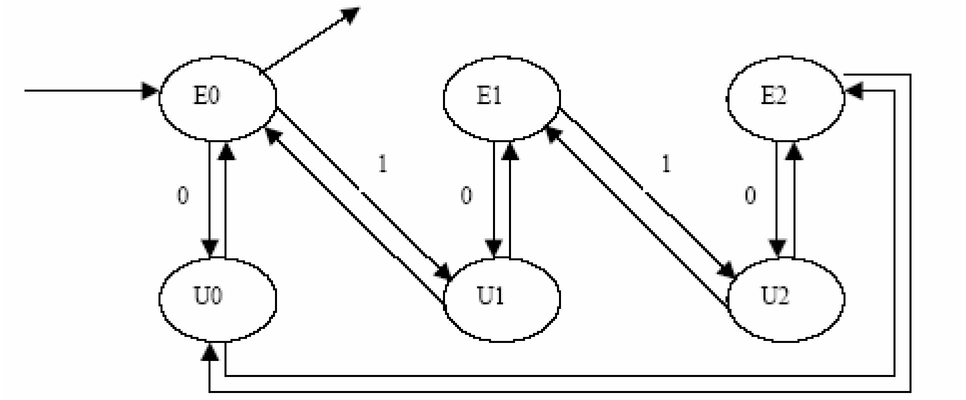
	1	2	3	4	5	6	7	8	Total
Grade	/xx	/xx	/xx	/xx	/xx	/xx	/xx	/xx	/100
Grader									

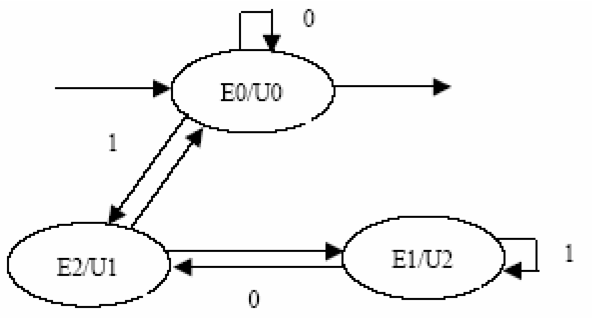
1. (xx points)

   A. The binary representation of a natural number is divisible by 3 when the number of 1's in even positions minus the number of 1's in odd positions is a multiple of 3 (where the rightmost, least significant, bit is numbered 0). More formally, if we let E be the number of 1's in even positions and U be the number of 1's in odd positions, then (E-U) mod 3 = 0. For example, 1407 has the binary representation 10101111111 for which E=6 and U=3, (6-3) mod 3=0, and thus it is divisible by 3.

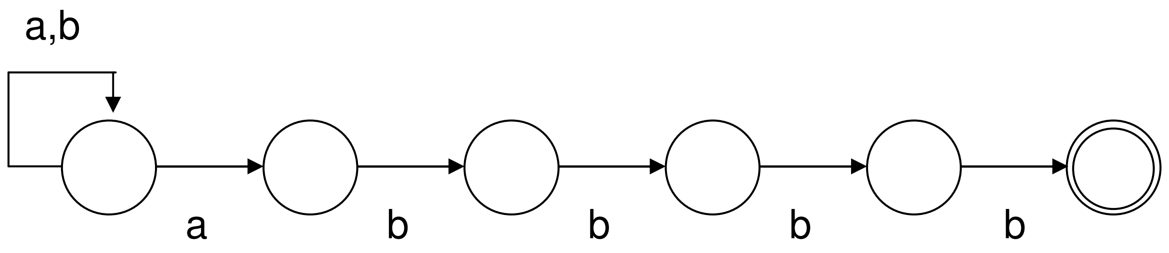
   Use this to construct a DFA that accepts strings of 0's and 1's such that the corresponding natural number is divisible by 3. That is, your DFA should accept any string such that (E-U) mod 3 = 0 and reject any other string. Assume that your DFA reads its input from right to left, that is with least significant bit first (or equivalently assume that the string is reversed before being passed to the DFA).

   For full credit you should create correct a DFA with the smallest possible number of states (which is 3), however a correct DFA with 6 states will receive substantial partial credit. Draw a diagram of the states and transitions of your DFA.





B. Consider the set of strings over the alphabet {a,b}. Construct an NDFA that accepts all strings over this alphabet that end with abbbb. Your NDFA should have the minimum number of possible states. Draw a diagram of the states and transitions of your NDFA.



2. (xx Points) Consider the problem of representing a directed graph by storing the set of vertices adjacent to each vertex. There are several ways of storing such a representation, including:

   1) a vector of lists
   2) vector of red-black trees
   3) a vector of splay trees
   4) a list of lists

   For each of the following problems, which of these four would be the best scheme to use in terms of the running time (considering both asymptotic time and constant factors):

   A) For a large number of randomly chosen pairs of vertices, check whether there is an edge between each pair.

   2

   B) Perform breadth first search starting from a specific vertex.

   1

   C) Determine the in-degree of a specific vertex.

   2

3. (xx Points) If you have items, such as natural numbers, that can be stored in an ordered set, and you need to do many lookups to see what numbers are in the set, when would you choose splay trees over red-black trees? In particular, briefly answer each of the following:

A) For what kinds of request patterns would splay trees be most appropriate?

Requests for the same item are clustered together in time.

B) For what kinds of request patterns would splay trees be least appropriate?

Requests come in random order.

C) For what kinds of applications are splay trees not very appropriate, even though the request pattern might suggest using them?

Real-time applications that cannot tolerate some lookups taking a long time.

4. (xx Points) Consider the following implementation of a queue as a circular mutable linked list.

```
datatype 'a ll = LL of 'a * ('a ll) ref | Null
type 'a q = (('a ll) ref * ('a ll) ref)
exception Empty

fun insert ((last,first):'a q, a:'a):unit =
  let
    val tmp = ref Null
    val v = LL(a, tmp)
  in
    case !last of
      Null =>
       (tmp := v; first := v; last :=v; ())
    | LL(b, l) =>
       (l := v; tmp := !first; last := v; ())
  end
```

In the following function for removing the head item of the queue, provide the missing code for each of the parts marked ***A***, ***B*** and ***C***.

```
fun remove ((last,first):'a q): 'a =
  case !first of
    Null => A
  | LL(b, next) =>
      let
        val LL(x,y) = !last
      in
        (if (next = y)
           then B
         else C;
         b)
      end
```

A) `raise Empty`

B) `(first:=Null; last:=Null)`

C) `(y:=(!next); first:=(!next))`

5. (xx Points)

A) What is the value of the expression below?

```
let
  val foo = ref (fn (x:int) => 0)
  fun f(x:int) =
    (foo := (fn (y:int) => x+y);
     if (x=0) then (!foo)(x)
     else (!foo)(x) + f(x-1) + (!foo)(x))
in
  f(3)
end
```

```
18
```

B) Draw the environment diagram corresponding to the following expression.

```
let
  val x = 2
  fun f(x) =
    if (x=0) then 1
    else f(x-1)+f(x-1)
in
  f(x)
end
```

6. (xx Points) Consider representing a list as an array. An array is initially created with one element. As elements are added to the list, if the array is full then a new larger array is created and the existing elements are copied into that array. Various schemes are possible for increasing the size of the array when it gets full. For each of these, state what the amortized running time is per item, when m items are added to a list. Use big-oh notation.

   A) Create a new array of double the size of the previous array.

   O(1), constant, amortized time per item.

   B) Create a new array 1.25 times the size of the previous array.

   O(1), constant, amortized time per item.

   C) Create a new array of with 100 more entries than the previous array.

   O(m), linear, amortized time per item.

7. (xx Points) Consider the following simple SML code

```
fun square (n:int) = n*n
fun double (n:int) = n+n

fun exp 0 = 1
  | exp n =
    if n mod 2 = 0 then
      square (exp (n div 2))
    else
      double (exp (n - 1))
```

A) Prove that for n>=0, exp(n) evaluates to 2^n. You must clearly state your induction hypothesis, base case and inductive step.

Base case, if n=0 the code returns 1, which equals 2^0.

Induction step. Assume exp(m)= 2^m for all m<n, show for n.

If n is even then the program returns square($2^{n \text{ div } 2}$) by the induction hypothesis, because n div 2 < n. Using the definition of square this is $2^{n \text{ div } 2} * 2^{n \text{ div } 2}$ which is 2^n.

If n is odd then the program returns double(2^{n-1}) by the induction hypothesis, because n-1 < n. Using the definition of double this is $2^{n-1}+2^{n-1}$ which is 2^n.

B) What is the asymptotic running time of exp(n) as a function of the magnitude of the number n? Carefully and concisely justify your answer.

O(logn) because there cannot be successive odd recursive calls, n-1 is even if n is odd, and thus n is halved on at least half the calls.

8. (xx Points) Consider the following concurrent programs, both of which have errors in the way(s) in which they synchronize or fail to synchronize the operation of multiple threads. For each program, briefly explain what is wrong, and what changes you would make to the code to fix it.

A) In class we considered an implementation of hash tables. The following code fragment inserts a key value pair into a given hash table.

```
fun insert(ht,k,v) =
  let
    val bucket = Int.mod(hash(k),Array.length(ht))
    val entries = Array.sub(ht,bucket)
  in
    case find(k,entries) of
      NONE => Array.update(ht,bucket,(k,v)::entries)
      | _ => Array.update(ht,bucket,
                          (k,v)::remove(k,entries))
  end
```

This code is not thread safe. If two threads try to insert into a hash table at the same time the results will be unpredictable. In one or two sentences explain how you would make this code thread safe. Your answer should be both precise and concise.

Make ht a synchronous variable and do CML.mTake before accessing the entries in the bucket.

B)

```
fun prog2() =
  let
    val s1 = SyncVar.mVarInit(1)
    val s2 = SyncVar.mVarInit(2)
    fun a(n: int) =
      let
           val va = SyncVar.mTake(s1)
           val vb = SyncVar.mTake(s2)
      in
           TextIO.print("A " ^ Int.toString(va) ^ " "
                        ^ Int.toString(vb) ^ "\n");
           SyncVar.mPut(s1,va-n);
           SyncVar.mPut(s2,vb+n);
           a(n)
      end
    fun b(n: int) =
      let
           val va = SyncVar.mTake(s2)
           val vb = SyncVar.mTake(s1)
      in
           TextIO.print("B " ^ Int.toString(va) ^ " "
^
                      Int.toString(vb) ^ "\n");
           SyncVar.mPut(s1,va+n);
           SyncVar.mPut(s2,vb-n);
           b(n)
      end
  in
    CML.spawn(fn() => a(3));
    CML.spawn(fn() => b(2));
    ()
end
```

This code can deadlock because the locks on s1 and s2 are acquired in opposite order. To fix it simply change the mTake calls in a and b to be in the same order.