Base case: a0 = 2  
 b0 = (3\*0) + 2  
 a0 = b0 = 2  
Assume: ak = bk

ak = ak-1 + 3

bk = 3k + 2

Verify: ak+1 = bk+1

ak+1 = ak + 3

ak+1 = (3k + 2) + 3

bk+1 = 3(k+1) + 2

3k + 3 +2 = 3k + 2 + 3

3k = 3k

k = k

Hence the statement is true for *n* = *k*+1, and   
{*an*} = {*bn*} iff *an* = *bn*, *n* ∈ ℕ0 where *a*0 = 2, *an* = *an*-1 + 3, and *bn* = 3*n*+2

1. Answers:
   1. No; it is possible to produce a string with an even-sized length. To close the set under concatenation, include the even-length strings.
   2. No; it is possible to have a string ending with α. To close the set, include the strings where *w* ends in α.
   3. **SameColour** lacks all the unique properties of an equivalence relation – it is not reflexive, symmetric or transitive
   4. The equivalence closure of **SameColour** is:   
      { (*apricot, orange*), (*avocado, lime*), (*lime, cucumber*), (*strawberry, tomato*), (*carrot, orange*), (*carrot, carrot*), (*avocado, avocado*), (*lime, lime*), (*tomato, tomato*), (*pear, pear*), (*apricot, apricot*), (*strawberry, strawberry*), (*orange, orange*), (*cucumber, cucumber*), (*orange, apricot*), (*lime, avocado*), (*cucumber, lime*), (*tomato, strawberry*), (*orange, carrot*), (*apricot, carrot*), (*carrot, apricot*), (*avocado, cucumber*), (*cucumber, avocado*) }
   5. 
   6. As the above picture shows, there are 4 equivalence classes in the closure: [*apricot, orange, carrot*], [*avocado, lime, cucumber*], [*strawberry, tomato*] and [*pear*].
   7. False; if *L1* = { **ε** } and *L2* =/= { **ε** }, **ε*L2* = *L2*ε or *L2* = *L2***
   8. True; *L1+* = *L1L1\** and *L2+* = *L2L2\**
   9. False; if *L1* = *L2* but *L1* and *L2* are infinite, then *L1 – L2* = { **ε** } and |*L1 – L2*| = 1 – in other words, the cardinality of the language is 1 and therefore finite.
   10. M = ({ q0, q1, q2,q3 }, { 0, 1 }, δ, q0, { q3 })  
       δ = { ((q0, 0)q0), ((q0, 1)q1), ((q1, 0)q2), ((q1, 1)q0), ((q2, 0)q0), ((q2, 1)q3), (q3, 0)q3,   
       (q3, 1)q3 }
   11. M = ( { q0, q1, q2, q3, q4, q5}, { a, b }, δ, q0, { q0 } )  
       δ = { ((q0,a)q1), ((q0,b)q2), ((q1, a)q4), ((q1, b)q3), ((q2, a)q3), ((q2, b)q4), ((q3, a)q0), ((q3, b)q5), ((q4, a)q0), ((q4, b)q0), ((q5, a)q5), ((q5, b)q5)} 
2. minDFSM:   
   classes = { [6, 7], [1, 2, 3, 4, 5] }  
   ((6, a), [1, 2, 3, 4, 5]) ((6, b), [6, 7]) ((7, a), [1, 2, 3, 4, 5]) ((7, b), [6, 7])  
     
   ((1, a), [1, 2, 3, 4, 5]) ((1, b), [6, 7]) ((2, a), [1, 2, 3, 4, 5]) ((2, b), [6, 7])   
   ((3, a), [1, 2, 3, 4, 5]) ((3, b), [1, 2, 3, 4, 5]) ((4, a), [6, 7]) ((4, b), [1, 2, 3, 4, 5])   
   ((5, a), [6, 7]) ((5, b), [1, 2, 3, 4, 5])  
     
   classes = { [6, 7], [1, 2], [3], [4, 5] }  
   ((1, a), [4, 5]) ((1, b), [6, 7]) ((2, a), [1, 2]) ((2, b), [6, 7])  
   ((4, a), [6, 7]) ((4, b), [4, 5]) ((5, a), [6, 7]) ((5, b), [4, 5])  
   ((6, a), [3]) ((6, b), [6, 7]) ((7, a), [3]) ((7, b), [6, 7])  
     
   classes = { [6, 7], [1], [2], [3], [4, 5] }  
   ((4, a), [6, 7]) ((4, b), [4, 5]) ((5, a), [6, 7]) ((5, b), [4, 5])  
   ((6, a), [3]) ((6, b), [6, 7]) ((7, a), [3]) ((7, b), [6, 7])

*M* does not contain the minimum number of required states. There are 5 equivalence classes; therefore, 2 states can be removed from the DFSM.

* 1. M = ({ q0, q1, q2, q3, q4, q5, q6 }, { a, b }, Δ, q0, q6)  
     Δ = 
  2. M = ({ q0, q1, q2, q3, q4, q5, q6 }, { a, b }, Δ, q0, q6)  
     Δ = 

1. eps(1) = { 1, 2 } eps(2) = { 2 } eps(3) = { 3 } eps(4) = { 4 }  
   eps(5) = { 4, 5 } eps(6) = { 5, 6 }

eps(s) = { 1, 2 }  
active states: ( **{ 1, 2 }, { 3, 4 }, { 4, 5, 6 }, { 2 }, { 4, 5 }, { 2, 4, 5}, { 4 }** )  
({ 1, 2 }, a) = eps({ 3, 4 }) = { 3, 4 } ({ 1, 2 }, b) = eps({ 5, 6 }) = { 4, 5, 6 }   
({ 3, 4 }, a) = eps({ 2 }) = { 2 } ({ 3, 4 }, b) = eps({ 5 }) = { 4, 5 }  
({ 4, 5, 6 }, a) = eps({ 2, 5 }) = { 2, 4, 5 } ({ 4, 5, 6 }, b) = eps ({ 5, 6 }) = { 4, 5, 6 }  
({ 2 }, a) = eps({ 4 }) = { 4 } ({ 2 }, b) = ∅  
({ 4, 5 }, a) = eps({ 2, 5 }) = { 2, 4, 5 } ({ 4, 5 }, b) = eps ({ 5, 6 }) = { 4, 5, 6 }  
({ 2, 4, 5 }, a) = eps({ 2, 4, 5 }) = { 2, 4, 5 } ({ 2, 4, 5 }, b) = eps ({ 5, 6 }) = { 4, 5, 6 }  
({ 4 }, a) = eps({ 2 }) = { 2 } ({ 4 }, b) = eps({ 5 }) = { 4, 5 }

Since ¬L(M) is the result where the accepting state does NOT contain states 2 and 5:  


* 1. standardise(*M*: FSM) =   
     1. No states are unreachable.  
     2. Start states are bound in loops – adding new start state with an **ε-transition.  
     3.   
     dfsmtoregex(*M*)**

**q1 rip:**

**R’(0,2) = R(0,2)** ⋃ **R(0,1) R(1,1)\* R(1,2) = a\*b  
R’(0,3) = R(0,3)** ⋃ **R(0,1) R(1,1)\* R(1,3) = 0  
R’(0,4) = R(0,4)** ⋃ **R(0,1) R(1,1)\* R(1,4) = 0  
R’(0,5) = R(0,5)** ⋃ **R(0,1) R(1,1)\* R(1,5) = a\***

**R’(2,3) = R(2,3)** ⋃ **R(2,1) R(1,1)\* R(1,3) = b  
R’(2,4) = R(2,4)** ⋃ **R(2,1) R(1,1)\* R(1,4) = 0  
R’(2,5) = R(2,5)** ⋃ **R(2,1) R(1,1)\* R(1,5) = aa\***

**R’(3,2) = R(3,2)** ⋃ **R(3,1) R(1,1)\* R(1,2) = 0  
R’(4,2) = R(2,4)** ⋃ **R(2,1) R(1,1)\* R(1,4) = 0   
R’(4,3) = R(2,4)** ⋃ **R(2,1) R(1,1)\* R(1,4) = 0  
R’(2,5) = R(2,5)** ⋃ **R(2,1) R(1,1)\* R(1,5) = aa\*  
R’(3,4) = R(3,4)** ⋃ **R(3,1) R(1,1)\* R(1,4) = 0  
R’(3,5) = R(3,5)** ⋃ **R(3,1) R(1,1)\* R(1,5) = ε  
R’(4,5) = R(4,5)** ⋃ **R(4,1) R(1,1)\* R(1,5) = ε**

**q2 rip:**

**R’(0,3) = R(0,3)** ⋃ **R(0,2) R(2,2)\* R(2,3) = a\*bb  
R’(0,4) = R(0,4)** ⋃ **R(0,2) R(2,2)\* R(2,4) = 0  
R’(0,5) = R(0,5)** ⋃ **R(0,2) R(2,2)\* R(2,5) = a\*** ⋃ a\*b **R’(3,4) = R(3,4)** ⋃ **R(3,2) R(2,2)\* R(2,4) = a  
R’(3,5) = R(3,5)** ⋃ **R(3,2) R(2,2)\* R(2,5) = ε   
R’(4,3) = R(4,3)** ⋃ **R(4,2) R(2,2)\* R(2,3) = 0  
R’(4,5) = R(4,5)** ⋃ **R(4,2) R(2,2)\* R(2,5) = ε**

**q3 rip:**

**R’(0,4) = R(0,4)** ⋃ **R(0,3) R(3,3)\* R(3,4) = a\*bbb\*a  
R’(0,5) = R(0,5)** ⋃ **R(0,3) R(3,3)\* R(3,5) = a\*** ⋃ a\*b ⋃ a\*bb**b\*  
R’(4,5) = R(4,5)** ⋃ **R(4,3) R(3,3)\* R(3,5) = 0**

**q4 rip:**

**R’(0,5) = R(0,5)** ⋃ R(0,4) R(4,4)\* R(4,5) = **a\*** ⋃ a\*b ⋃ a\*bb**b\***

**Hence, the regex is a\*** ⋃ a\*b ⋃ a\*bb**b\*.**

* 1. standardise(*M*: FSM) =   
     1. No states are unreachable.  
     2. Start states are bound in loops – adding new start state with an **ε-transition.  
     3. dfsmtoregex(*M*)  
     q1 rip:**

**R’(0,2) = R(0,2)** ⋃ **R(0,1) R(1,1)\* R(1,2) = b\*a  
R’(0,3) = R(0,3)** ⋃ **R(0,1) R(1,1)\* R(1,3) = 0  
R’(0,4) = R(0,4)** ⋃ **R(0,1) R(1,1)\* R(1,4) = 0**

**R’(2,3) = R(2,3)** ⋃ **R(2,1) R(1,1)\* R(1,3) = b  
R’(2,4) = R(2,4)** ⋃ **R(2,1) R(1,1)\* R(1,4) = 0**

**R’(3,2) = R(3,2)** ⋃ **R(3,1) R(1,1)\* R(1,2) = bb\*a  
R’(3,4) = R(3,4)** ⋃ **R(3,1) R(1,1)\* R(1,4) = a**

**q2 rip:**

**R’(0,3) = R(0,3)** ⋃ **R(0,2) R(2,2)\* R(2,3) = b\*ab  
R’(0,4) = R(0,4)** ⋃ **R(0,2) R(2,2)\* R(2,4) = 0   
R’(3,4) = R(3,4)** ⋃ **R(3,2) R(2,2)\* R(2,4) = a**

**q3 rip:**

**R’(0,4) = R(0,4)** ⋃ **R(0,3) R(3,3)\* R(3,4) = b\*aba**

**Since the accepting state is bound in a reflexive loop, the regex is b\*abaa\*b\*.**