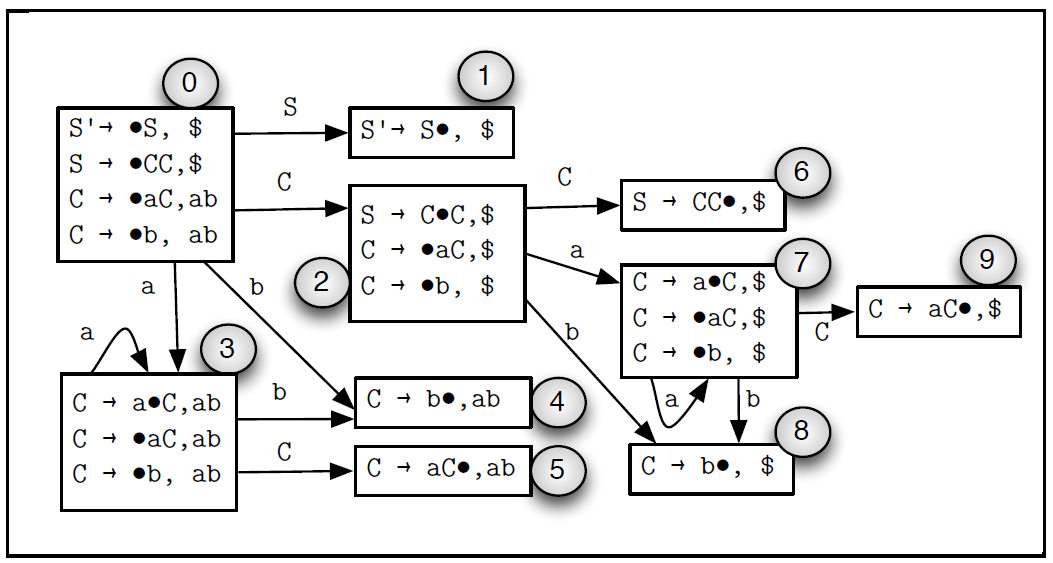
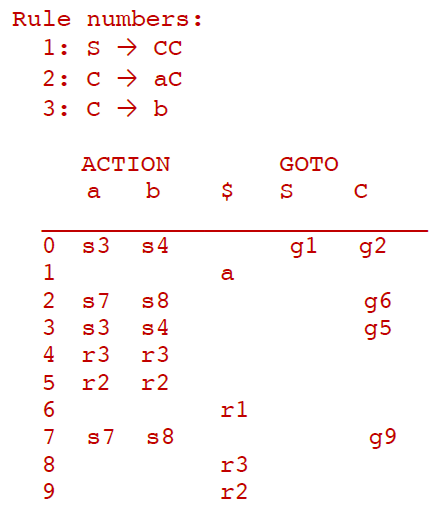
**‘\1a** (Answers taken from Dulay’s materials)

**i)**



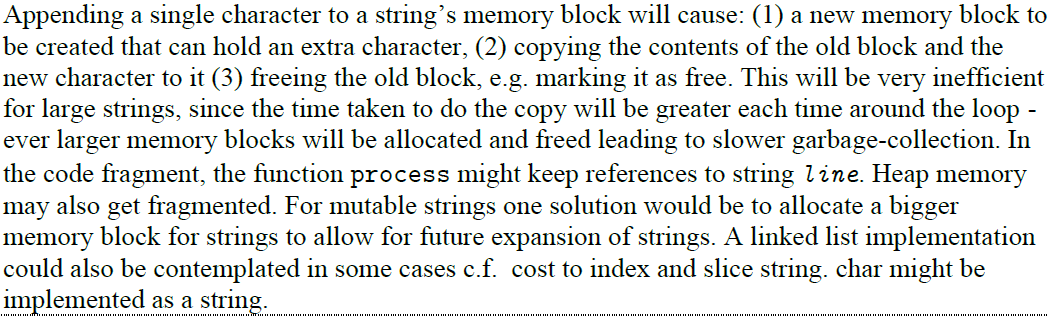
**ii)**



**iii) & iv)**



**1b**



**2a**

**i)**

No, a path exists from S2 to S8 without going through S4, which is the case when the while loop condition fails to hold from the start.

**ii)**

Yes, all paths of execution of the function must begin with S2. As such all paths that reach S4 must have gone through S2.

**iii)**

Yes. y is used in S6, and so belongs to its LiveIn set. Since S6 is a successor of S5, the LiveOut set of S5 includes all variables in the LiveIn set of S6.

**iv)**

Yes. S8 uses z, and is a successor of S3. S3 does not redefine z, and so z belongs to LiveIn(S3). Since S3 is an immediate successor of S2, z then belongs to LiveOut(S2).

**v)**

No. S6 kills the original definition of z from S2 with its own definition of z.

**2b**

if (x > 0) {

x = 1;

} else {

x = 0;

}

while (x == 0) {

...

}

**2c**

transExp :: Exp -> Operand

transExp (Const i)

= ImmNum i

transExp (Var x)

= Abs x

transStat :: Stat -> [Instr]

transStat (Assign name exp)

= [Mov (transExp exp) (Abs name)]

transStat (Switch var cases)

= concatMap transCond casesWithLabels ++

[Bra endLabel] ++

concatMap transBody casesWithLabels ++

[Define endLabel]

where

endLabel :: String

endLabel = show 0

-- Uses labels from 1 to n, where n is the number of cases

casesWithLabels :: [(Int, (Int, Stat))]

casesWithLabels = zip [1..] cases

-- Generates the comparison and branching instructions for each

-- case

transCond :: (Int, (Int, Stat)) -> [Instr]

transCond (label, (i, \_))

= [Cmp (Abs var) (ImmNum i),

Beq (show label)]

-- Generates the labels as instructions, and the case body by

-- recursively calling transStat

transBody :: (Int, (Int, Stat)) -> [Instr]

transBody (label, (\_, stat))

= [Define (show label)] ++ transStat stat ++ [Bra endLabel]

translate :: [Stat] -> [Instr]

translate = concatMap transStat