1. Write regular expressions for the following languages over the alphabet $\Sigma = \{a, b\}$:

   (a) All strings that do not end with aa.

   $$\epsilon + a + b + (a+b)^*(ab + ba + bb)$$

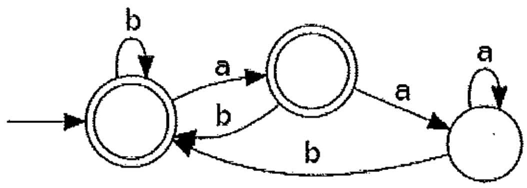
   (b) All strings that contain an even number of b's.

   $$a^*(ba^*ba^*)^*$$

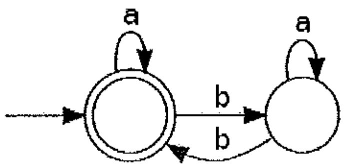
   (c) All strings which do not contain the substring ba.

   $$a^*b^*$$

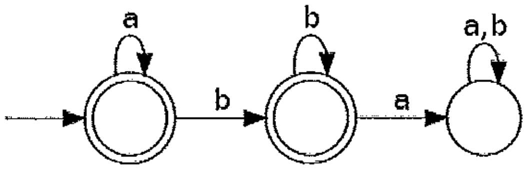
2. Draw DFAs for each of the languages from question 1. None of your DFAs may contain more than 4 states.



(a)



(b)



(c)

Page to be used for answers to Questions 2, 3 and 4

Answer to Question 2

(a) The LL(1) parsing table contains multiple entries for the pair: (S, b)

The entries are: $S \rightarrow Sa$ and $S \rightarrow b$

(b) A grammar which is unambiguous, left-factored, not left-recursive and also not LL(1) is:

$S \rightarrow B \mid C$

$B \rightarrow ab$ (others are possible)

$C \rightarrow ac$

Answer to Question 3

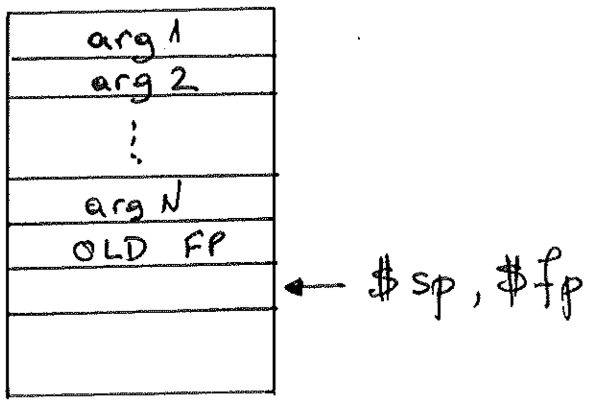
1. $First(A) = \{x, y\}$ is correct for grammars: 2
2. $Follow(A) = \{\$, x\}$ is correct for grammars: 1, 2, 3
3. $Follow(B) = \{\$, x, y\}$ is correct for grammars: 3
4. $First(C) = \{y\}$ is correct for grammars: 1, 2
5. $Follow(C) = \{\$, x\}$ is correct for grammars: 1, 2, 3

Answer to Question 4

1. With call-by-value and lexical scope the program prints: 10
2. With call-by-value and dynamic scope the program prints: −90
3. With call-by-reference and lexical scope the program prints: 20
4. With call-by-reference and dynamic scope the program prints: 0

QUESTION 5

(a)



(b)

b) Write the code for return (the ... in the above code). To return you must use the instruction `jr $ra` (jump to the address in register $ra). Recall that the result value must be in $a0 and the calling function pops the arguments.

```
lw $fp 4($sp)
addiu $sp $sp 4
jr   $ra
```

(c)

c) Write the code for function call. To perform the actual call use the instruction `jal f` (jump to the address of function `f` and save the return address in register $ra).

```
cgen( f(e1, e2, ..., en) ) =

  sw  $ra 0($sp)       # Save the return address, since
  addiu $sp $sp -4     # cgen(ei) and jal below can clobber it
  cgen(e1)             # Eval args in order 1 -> n
  sw $a0 0($sp)        # Push them on stack
  addiu $sp $sp -4
  cgen(e2)
  sw $a0 0($sp)
  addiu $sp $sp -4
  ...
  cgen(en)
  sw $a0 0($sp)
  addiu $sp $sp -4
  jal   f             # function call
  addiu $sp $sp 4*n   # Pop the arguments
  lw $ra   4($sp)     # Reload the saved $ra
  addiu $sp $sp 4     # fix $sp
```

QUESTION 6

(a) + (d)

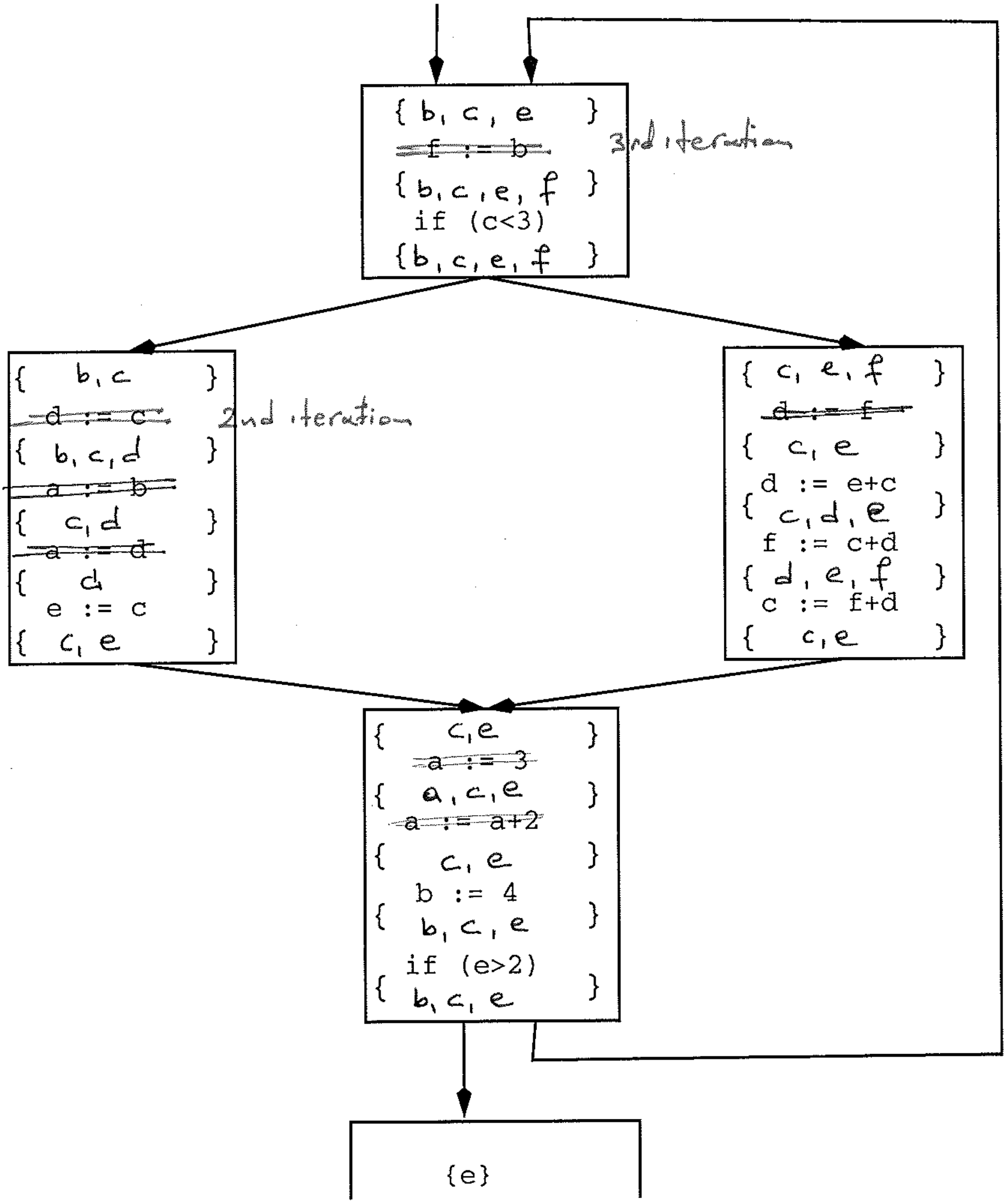
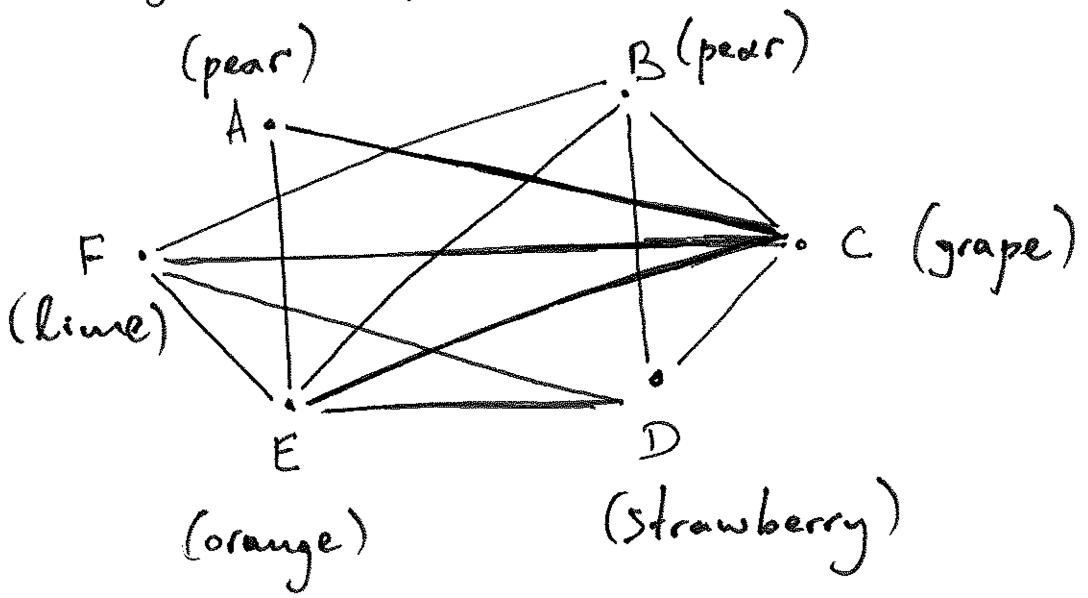


Figure 1: Control flow graph to be used for the answers to Questions 6a and 6d.

(b) Purely based on the information in the answer to (a), we need at least 4 colours since there are portions of the program in which four variables are simultaneously live.

(c) A register interference graph is



Note there are 5 colours used here, which is different than the answer to (b) which was a "lower" bound answer.

(d) See previous page; after that, liveness needs to be recomputed for the answer to (e).

(e) Now we need only 4 registers instead of 5.

QUESTION 7

The most important reason to take "KT1" is so that you understand how programs are executed in modern CPUs and the trade-offs that are involved.