NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER I (2008-2009) EXAMINATION FOR

CS4212 COMPILER DESIGN

December 2008 Time Allowed: **2 Hours**

INSTRUCTIONS TO CANDIDATES

1) This examination paper contains **FIVE** questions and comprises **SIXTEEN** printed pages.
2) Answer **ALL** the questions in the spaces provided.
3) This is an **OPEN-BOOK** examination.
4) The maximum mark of this paper is **100%**.
5) The questions are **not** arranged according to the order of increasing difficulty.

Matriculation Number : _____

For Examiner's Use Only			
Question	**Maximum**	**Marks**	**Check**
Q1(a,b)	**12**		
Q1(c,d)	**16**		
Q1(e)	**7**		
Q2(a)[a,b]	**11**		
Q2(b)	**10**		
Q3	**7**		
Q4(a)	**7**		
Q4(b)	**7**		
Q5(a)	**8**		
Q5(b)	**7**		
Q5(c)	**8**		
TOTAL :	**100**		

This space is intentionally left blank.

1. Consider the following CFG:

$S \rightarrow A\,\$$

$A \rightarrow B$

$A \rightarrow B\,\mathbf{1}\,A$

$B \rightarrow \mathbf{0}$

(a) **[6%]** Generate the set of items (specifically, LR(0) items) from the above rules. The first item is given as $S \rightarrow \cdot A\,\$$

Ans:

(b) **[6%]** Build the set of closures from the set of items.

Ans:

(c)[**8%**] Build (by drawing) a DFA out of the set of closures.

Ans:

(d) **[8%]** Construct a parse table from the DFA built.

Ans:

	0	**1**	**$**	*A*	*B*
1					
2					
3					

(e) **[7%]** There exists some conflict in the parse table you have just built. Identify at least one such conflict and explain clearly how it can be resolved.

Ans:

2. Given a CFG *G*, for any non-terminal *A* in *G*, we say that *A* is **nullable** if and only if it can derive a null string (i.e., empty string, ε). Moreover, we say that the **first-set** associated with *A* is the set of tokens (including the empty string) that can begin a string derivable from *A*. We say that the **follow-set** associated with *A* is the set of tokens (excluding the empty string) that can come after *A* in a sentential form; and if *A* can appear last in a sentential form, then its **follow-set** includes $, representing the end-marker. (Reference: page 220 of the textbook, slides 7 – 12 of the lecture note w6.pdf.)
Consider the following CFG ζ containing three non-terminals, ***S***, ***T***, ***E***:

S → **0** ***E*** **1** ***T S*** | ***T T*** | **2**

T → **3** ***S*** | ε

E → **4**

(a) Answer the following questions pertaining to the CFG ζ :

a. **[3%]** Which of these non-terminals are nullable?

Ans:

b. **[8%]** What are the first-sets and follow-sets associated with each of the non-terminals?

Ans:

	First-set	Follow-set
E		
T		
S		

(b) **[10%]** The definition of the CFG ζ is repeated below for easy reference:

$S \rightarrow 0E1TS \mid TT \mid 2 \qquad T \rightarrow 3S \mid \varepsilon \qquad E \rightarrow 4$

Given a parse tree generated from ζ, we associate each node in the tree with three attributes:

i) *Nullable*: *A.Nullable returns* **True** if A derives to empty string

ii) *first*: *A.first* returns the first token derivable from A

iii) *follow*: *A.follow* returns the token following immediately from the derivation of A.

Provide a syntax directed definition of these three attributes [*Hint*: *There is no need to follow the left-to-right ordering in computing attributes.*]

Ans:

PRODUCTION	SEMANTIC RULES
$S \rightarrow 0E1TS'$	
$S \rightarrow TT'$	

Additional Working space.

PRODUCTION	SEMANTIC RULES

3. [7%] Determine the type of the following expression:

fun f(x) = ((fun g(y) = y) x)

Ans:

4. Some languages, like PL/I, will coerce a boolean value into an integer, with **True** identified with **1** and **False** with **0**. For example, **3 < 4 < 5** is grouped **left associatively**. Moreover, **(3 < 4)** has the value **True**, which will be coerced to **1**, and **1 < 5** has the value **True** (rather than **1**).

(a) **[7%]** Write a small CFG for accepting expressions of the above form, such that:

a. Only "<" operation is available, and the "<" operator is left associative.

b. There can be finitely many "<" in the sentence

c. All the operands are either "<" operations or identifiers denoted by token **id**.

Ans:

(b) [7%] Write translation rules for expressions defined by your CFG. Your translation rules will produce three-address code, which will also produce code that coerces Boolean values to integers when necessary. The format of the coercion instruction is given as follows:

```
X := (int) Y
```

Ans:

5. This question is about register allocation.

(a) **[8%]** Consider the following block of code very similar to three-address code:

```
g := j + 4
h := u * 10
y := g - h
e := arr[j + 8]
a := arr[j + 12]
b := arr[y]
c:= e +8
d := c
u := a + 4
j := b * 3
```

Suppose that variables d, j, and u are expected to be live at the end of the block. Perform a live variable analysis to determine the sets of live variables before and after each statement. Write your answer in the table provided below.

Statements	Live variable information
g := j + 4	
h := u * 10	
y := g - h	
e := arr [j+8]	
a := arr[j+12]	
b := arr[y]	
c := e + 8	
d := c	
u := a + 4	
j := b * 3	
	d, j, u

(b) [7%] Given the following sets of variables, where variables in each set are live simultaneously.

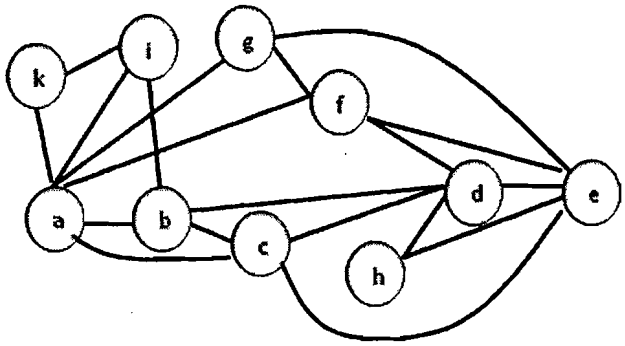
{x,y,f}, {x,y,z}, {c,y,z}, {u,y,z}, {u,v,z}, {u,v,j}

Assuming that all variables used are symbolic registers.

Generate a register-interference graph from these sets.

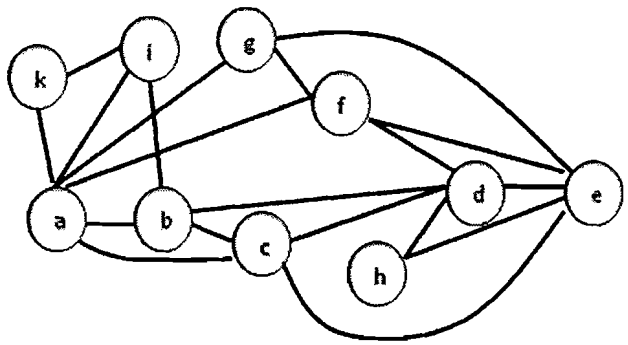
Ans:

(c) **[8%]** Consider the following register-interference graph,

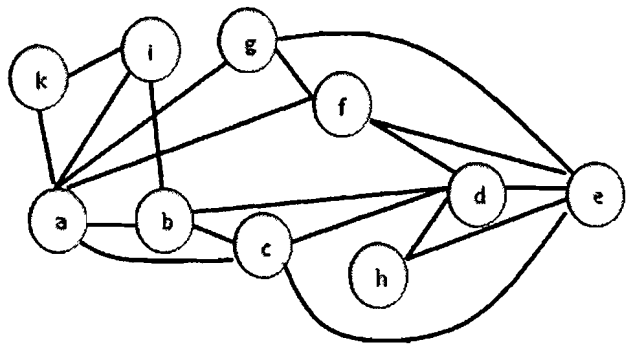


What is the minimum number of registers required to be allocated in order to ensure that there is no spilling effect in register allocation? Support your answer with a detailed labeling of physical registers (e.g., R1, R2, ...) to each node in the register-interference graph. **Write your answer at the end of the following page.**

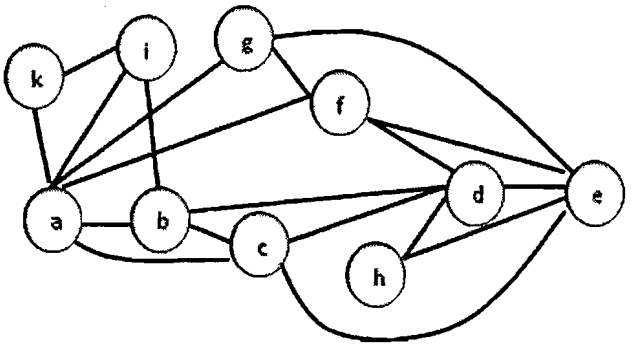
[*Working draft 1*]



[*Working draft2*]



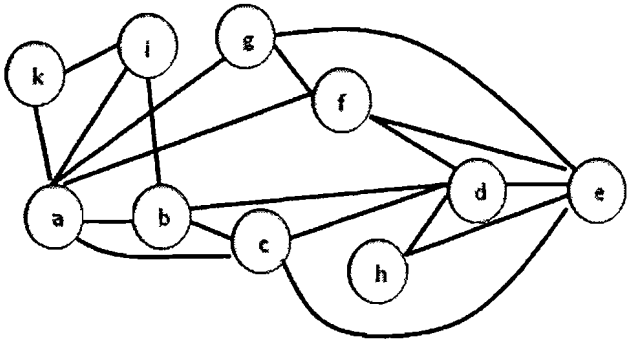
[*Working draft 3*]



[Your answer]

The minimum number of registers required is: _____

The final register allocations are as follows:



—THE END OF THE PAPER—

This space is intentionally left blank.