

**NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING
EXAMINATION FOR
Semester 1, 2010/2011**

CS4212 — Compiler Design

November 2010

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. The examination paper contains **FIVE (5) questions** and comprises **ELEVEN (11) pages**.
2. The maximum attainable score is 50.
3. All questions must be attempted for the maximum score to be attained.
4. This is an OPEN BOOK exam.
5. Write all your answers in the space provided in this booklet.
6. Please write your matriculation number below.

MATRICULATION NUMBER: _____

(this portion is for the examiner's use only)

Question	Marks	Remark
Q1		
Q2		
Q3		
Q4		
Q5		
Total		

Question 1 [10 marks]**Syntax Analysis**

Consider the following Prolog op declarations.

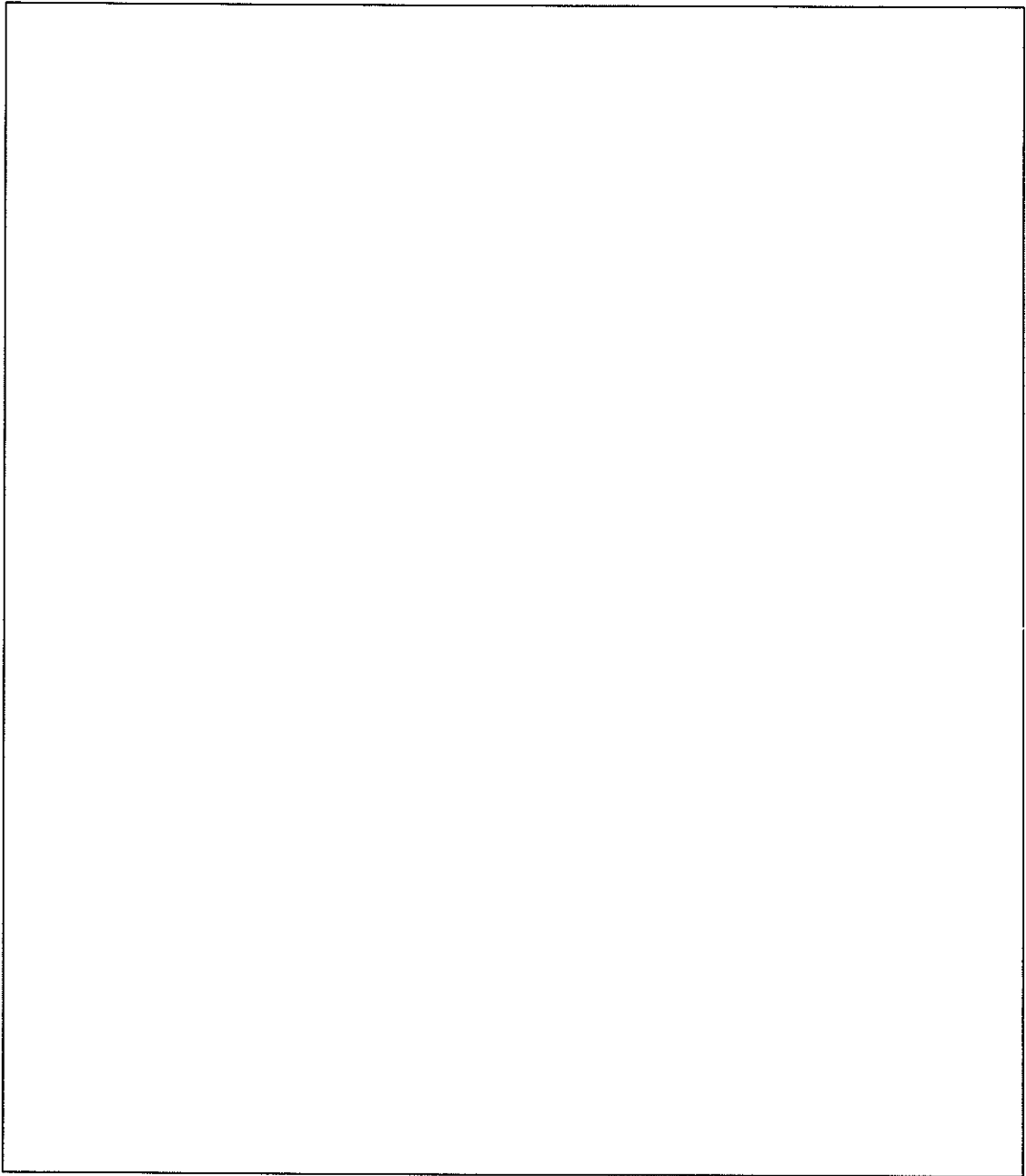
```
:- op(200, fy, +) .  
:- op(200, fy, -) .  
:- op(200, xfy, ^) .  
:- op(400, yfx, *) .  
:- op(400, yfx, /) .  
:- op(500, yfx, +) .  
:- op(500, yfx, -) .  
:- op(700, xfx, =) .
```

A. Assume that the Prolog system has no other operator declarations (that is, the usual predefined operator declarations have been, by some magic, retracted). Devise a non-ambiguous context free grammar that generates the language of expressions defined by the above declarations. Your grammar rules must reflect the precedence and associativity of the defined operators, and allow for parentheses, whose use is allowed by default in Prolog terms. [3 marks]

B. Provide left-most and right-most derivations using your grammar rules, for the following string:

$a = x*y/b^{\wedge}(c+d)^{\wedge}(e-f*g+h)$

[4 marks]



- C. Convert the grammar you provided in part A into one that is not left-recursive. [3 marks]

Question 2 [10 marks]

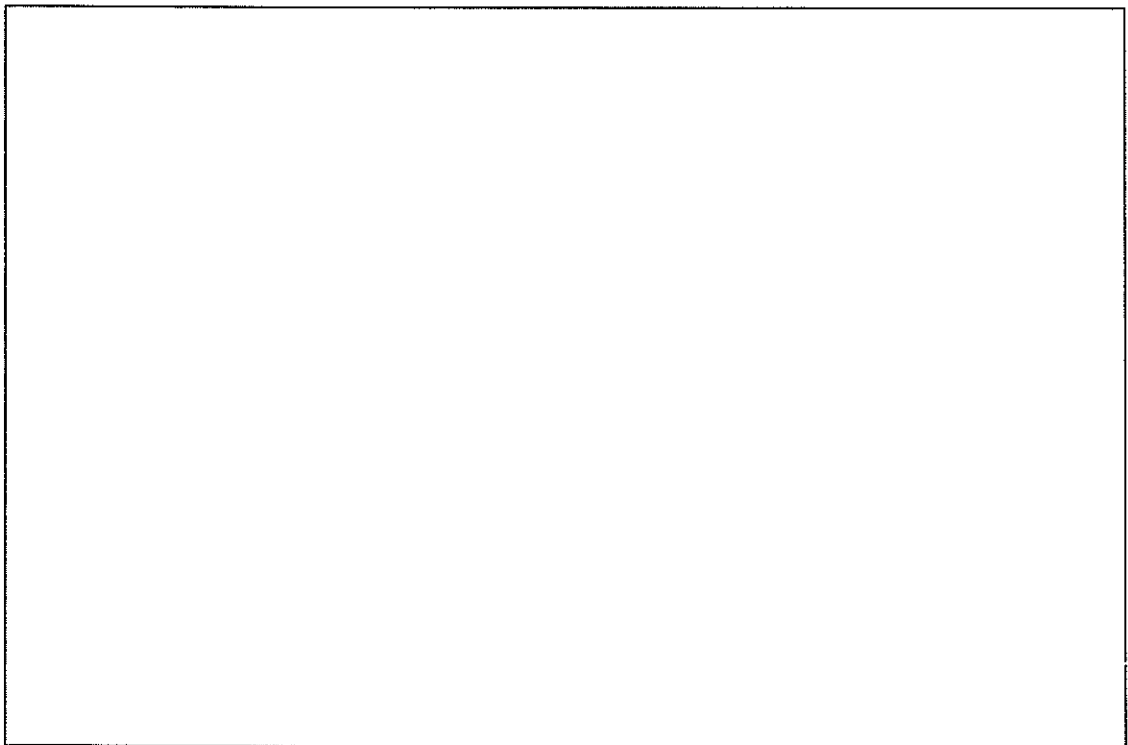
Lexical Analysis

Consider the language of Prolog clauses with the following restrictions:

- at most two levels of nested brackets;
- no lists
- no infix/prefix/postfix operators

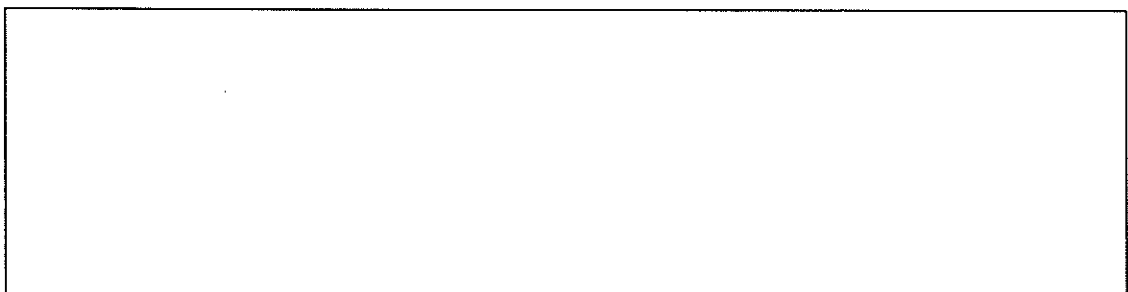
A. Devise a state diagram that accepts this language.

[5 marks]



B. Devise a regular expression that accepts this language. In your specification, you may use the regular expression syntax extensions available in Lex.

[5 marks]



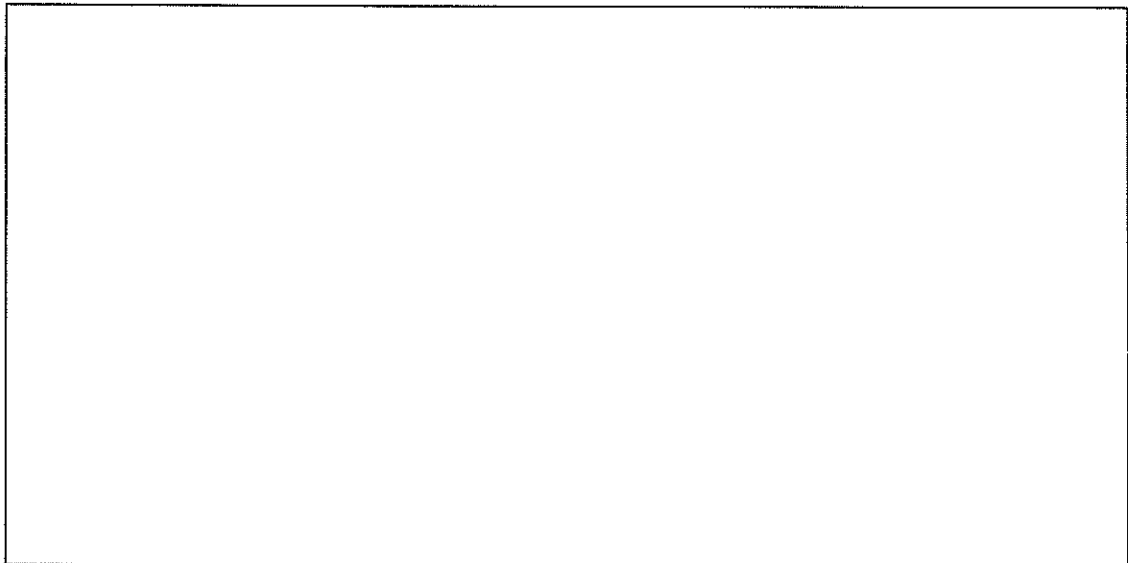
Question 3 [10 marks]

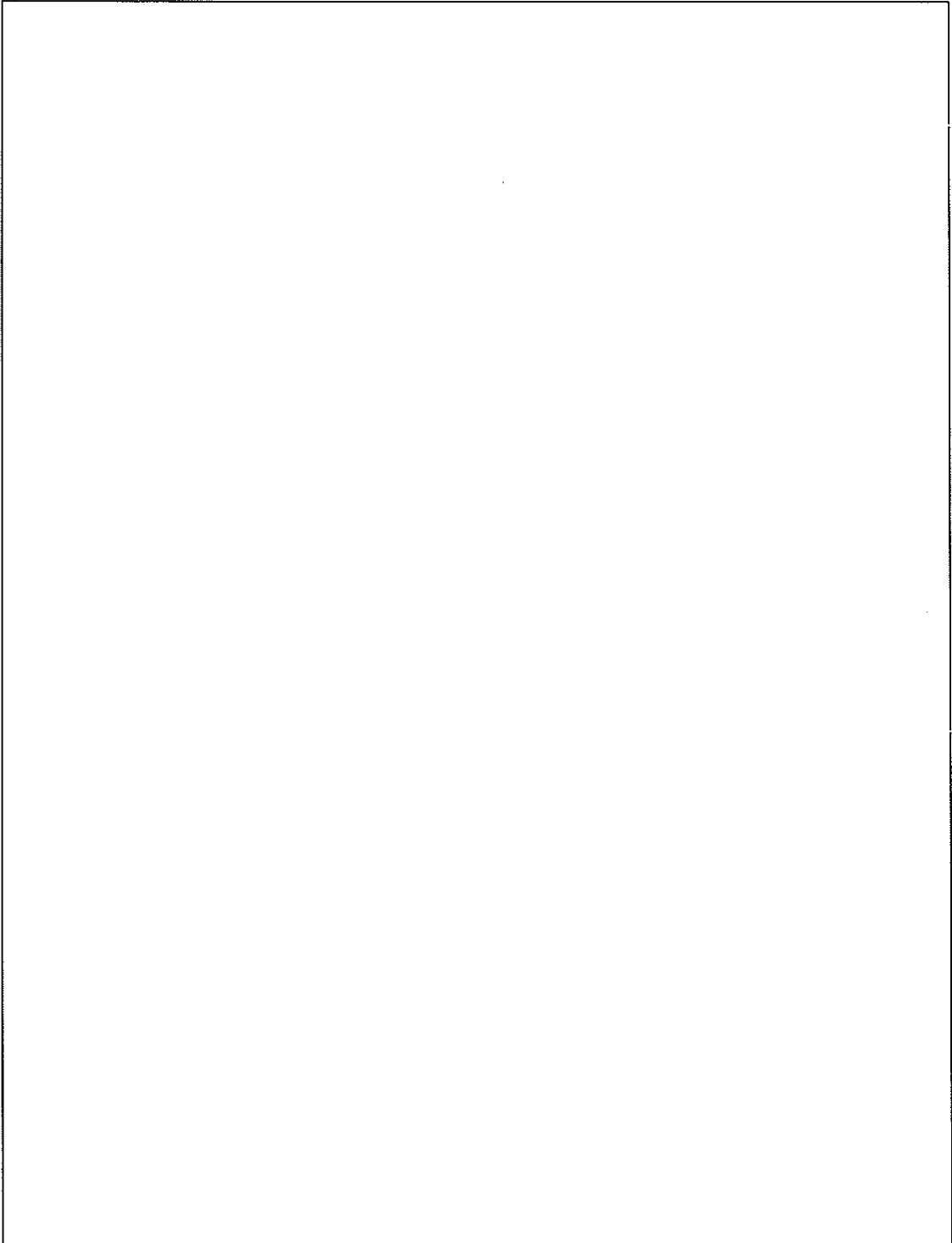
In a language like Python, the methods defined by a particular instantiation of an object may be changed at runtime. A simple C++ -like example taking advantage of this functionality is shown below. Discuss how a compilation scheme could make this feature possible. Showcase your compilation scheme by translating the code below into C.

```
class Hello {  
    void print() {  
        printf("hello ");  
    }  
}  
  
void printWorld() {  
    printf("world\n");  
}  
  
void main() {  
    Hello o = new Hello ;  
    o.print();  
    o.print = printWorld ;  
    o.print();  
}
```

Output of this code:

hello world





Question 4 [10 marks]

The compiler given in the Prolog file `03.pl` must be enhanced with a profiling capability. Specifically, the generated code must be able to count the number of both conditional and non-conditional jumps that are performed through execution; after running the code, this number must be available in the special register `gotoCount`. Explain in detail the changes that you would make to the compiler (i.e. the `compile` predicate) in order to implement this capability. Indicate which rules you would replace, and provide the rules that you would use as replacement.

Question 5 [10 marks]

Consider the following C++ code.

```
int f(int a, int b) {
    if(b==0) throw "Division by zero" ;
    return a/b ;
}

int g(int a, int b) {
    return a+f(a,b) ;
}

int h(int a, int b) {
    try {
        return g(a,b) ;
    } catch ( char * s ) {
        printf("Exception caught: %s\n", s) ;
        return 0 ;
    }
}

void main(int argc, char ** argv) {
    h(atoi(argv[1]),atoi(argv[2])) ;
}
```

Convert this program into C using the exception translation scheme discussed in Lecture 8.



— END OF PAPER —