1)

a) The following regular expression recognises certain strings consisting of the letters a, b and c:

```
a* (b|(ac))*
```

i) For the following 5 strings, indicate whether or not they are recognised by the above regular expression:

```
aacaac, abbac, bacbb, aabbabcbac, abacbbacb
```

[10 marks]

ii) Show three more strings that are recognised by the above expression.

[10 marks]

iii) Show two more strings consisting of the letters a, b and c that are ***not*** recognised by the above regular expression.

[5 marks]

b)

i) Explain why left-recursion must be eliminated from grammar productions which are to be used in construction of a recursive-descent parser.

[5 marks]

ii) Write down a general rule for rewriting left-recursive grammar productions to equivalent right-recursive grammar productions.

[10 marks]

iii) Use the general rule from part ii) to rewrite the following productions to be right-recursive:

```
A → A + B
A → C
B → B * C
B → C
C → ( A )
C → integer
```

[30 marks]

c) Consider the following Java class:

```
class Ninja {
  String name; String a;
  public void talk(int times, String topic) {
    System.out.println(topic);
    int a = 5;
    int b = times;
    System.out.println(a+b); System.out.println(times);
    System.out.println(name); System.out.println(a);
  }
}
```

Given an initial environment σ_0, derive the type binding environments for the method at each use of an identifier and indicate where type lookups will occur.

[30 marks]

[Total: 100 marks]

2)

a)

i) Explain what it means for a context-free grammar to be *ambiguous*.

[5 marks]

ii) From your explanation from i) above, show that the balanced parentheses grammar is ambiguous using the shortest string that will illustrate the ambiguity:

S → S + S
S → S – S
S → (S)
S → int

[15 marks]

b) The reference manual for a MiniJava-like programming language contains the following grammar for a *for* statement:

Statement → **for (***Statement* **;** *Exp* **;** *Statement***)** *Statement*

i) Sketch a possible abstract syntax for the *for* statement.

[10 marks]

ii) Show how semantic actions in a grammar for a parser-generator such as JavaCC can be used to produce abstract syntax trees for the *for* statement.

[25 marks]

iii) Informally describe an appropriate type check for the *for* statement.

[10 marks]

iv) Suppose a compiler for a MiniJava-like language that includes a *for* statement translates all statements and expressions into intermediate code (e.g. intermediate representation (IR) trees). Outline the intermediate code that might be generated in translation of the *for* statement.
You may wish to use a simple example to explain your translation, e.g.:

```
for( i=0; i<j; i=i+1 )
{ x=x+1; System.out.println(y); }
```

You can assume that the expression tree for any variable `v` is simply `TEMP v`. Do not show translations for the body of the example *for* statement (in braces in this example `{...}`).

[35 marks]

[Total: 100 marks]

3)

a) Why do many programming language implementations require a memory model that implements a runtime stack? Explain in detail how a stack frame is pushed to the stack, and removed from the stack, during program execution.

[30 marks]

b) Some programming language implementations avoid in some circumstances the need to pass parameters via a stack frame. Outline what these circumstances might be and why passing via the stack frame might be avoided. Also, outline situations where the use of a stack frame to pass parameters cannot usually be avoided.

[25 marks]

c) Suppose that a compiler translates a MiniJava-like language to an intermediate representation (for example IR trees) that will include the calculations required to address variables in stack frames. Draw or write down the intermediate representation required to access a local variable declared in a method. Explain your answer.

[20 marks]

d) Explain the difference between *caller-save* and *callee-save* registers. Study the following methods and suggest for each whether a caller-save or callee-save register is appropriate for variable `x`. Explain your answers.

```
void f (int i) { int x; x=i+1; a(i); b(i+1); }

int g (int j) { int x; x=x*j; b(); return x; }
```

[25 marks]

[Total: 100 marks]