1. (a) What is an ambiguous grammar? Show that the following grammar is ambiguous:

$$
\begin{array}{lll}
S & \rightarrow & \textbf{if } E \textbf{ then } S \\
  & | & \textbf{if } E \textbf{ then } S \textbf{ else } S \\
  & | & \textit{other}
\end{array}
$$

Here *other* stands for any other statement *S*. [15]

(b) The reference manual for a MiniJava-like programming language contains the following grammar rule for an if-statement:

$$\textit{Statement} \rightarrow \textbf{if} \ (\ \textit{Exp} \ ) \ \textit{Statement} \ \textbf{else} \ \textit{Statement}$$

i. Sketch a possible abstract syntax for the if-statement. [10]

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 if-statement. [25]

iii. Informally describe an appropriate typecheck for the if-statement. [10]

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

```
if (a < b) c = a; else c = b;
```

You can assume that the expression tree for any variable `v` is simply `TEMP v`. [40]

2. (a) The following regular expression recognises certain strings over the alphabet $\{a, b, c\}$

$$a(b|(bc))*\, c*$$

Indicate which of these five strings are recognised by the above regular expression:

acc, $abac$, a, $abcbcbccc$, $abbbccbc$

Also, show three more strings that are recognised by the above expression. Finally, show two more strings consisting of the letters a, b and c that are *not* recognised by the above regular expression. [25]

(b) Write a regular expression that recognises strings over the alphabet $\{a, b, c\}$ where there is an even number of a's. [20]

(c) Explain why left-recursion must be eliminated from grammar productions which are to be used in construction of a recursive-descent parser. Write down a general rule for rewriting left-recursive grammar productions to be right-recursive and use it to rewrite the following productions to be right-recursive:

$$\begin{array}{lll} S & \rightarrow & (\, L\,) \mid a \\ L & \rightarrow & L\, ,\, S \mid S \end{array}$$

[35]

(d) Consider the following Java class:

```
class A {
 String a; int c;
 public void f(int b, String c) {
   System.out.println(c);
   int d = 3;
   int a = b;
   System.out.println(a+d); System.out.println(b);
   System.out.println(c); System.out.println(d);
 }
}
```

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. [20]

3. (a) What programming language feature leads to the need for an implementation model that includes stack frames? Explain your answer. Explain in detail how a stack frame is pushed to the stack, and removed from the stack, during program execution. [30]

   (b) 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]

   (c) Explain why registers might be used for parameter passing and suggest situations where passing in registers is particularly appropriate. Outline situations where it is necessary for the code generated for a procedure or method to write registers to the stack. [25]

   (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.

```
int f (int a) { int x; x=a+1; g(); h(x); return x+2; }

void p (int y) { int x; x=y+1; q(y); q(2); }
```

[25]