1. The reference manual for a MiniJava-like programming language contains the following grammar rule for a do-while statement:

$$\textit{Statement} \rightarrow \textbf{do}\ \textit{Statement}\ \textbf{while}\ (\textit{Exp})\ ;$$

   (a) Write down or draw a possible abstract syntax for the do-while statement. [10]

   (b) Show how semantic actions in a grammar for a parser-generator such as JavaCC can be used to produce abstract syntax trees for the do-while statement. [25]

   (c) Informally describe an appropriate typecheck for the do-while statement. [10]

   (d) Suppose a compiler for a MiniJava-like language translates all statements and expressions into intermediate code (eg intermediate representation (IR) trees).

      i. Draw or write down the intermediate representation required to access a local variable declared in a method. Explain your answer. [20]

      ii. Suppose the MiniJava-like language includes a do-while statement. Outline the intermediate code that might be generated in translation of the do-while statement. You may wish to use a simple example to explain your translation, eg:

```
do {
  x = i*i; System.out.println (x); i = i+1;
} while (i < j) ;
```

      You can assume that the expression tree for any variable `v` is simply `TEMP v`. You need not show translations for the body of the example do-while statement (in braces in this example `{...}`). [35]

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

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

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

cbc, *abbc*, *c*, *abcbcbccc*, *ccbbcbcc*

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. [30]

(b) Consider the following grammar for strings of balanced parentheses:

S	→	*S S*
S	→	(*S*)
S	→	( )

Explain what it means for a context-free grammar to be ambiguous. Using your explanation, show that the balanced parentheses grammar is ambiguous using the shortest string that will illustrate the ambiguity. [30]

(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 for an arithmetic expression grammar to be right-recursive:

E	→	*E* + *T*
E	→	*T*
T	→	*T* * *F*
T	→	*F*
F	→	(*E*)
F	→	`integer`

[40]

3. (a) i. State two reasons why many programming language implementations require a memory model that implements a runtime stack? [10]

ii. Explain in detail how a stack frame is pushed to the stack, and removed from the stack, during program execution. [25]

(b) i. 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. [15]

ii. Explain three situations where the use of a stack frame to pass parameters cannot usually be avoided. [15]

(c) i. Explain the difference between *caller-save* and *callee-save* registers. [10]

ii. 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(x); return x+2; }

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

[25]