Question 1 *[30%]*

(i) The C programming language defines the lexical syntax of floating constants as follows. A floating constant consists of an integer part, a decimal point, a fraction part, an `E` and an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the `E` and the exponent (not both) may be missing. Give a regular expression that captures this set of strings. *(10%)*

(ii) Give a deterministic finite automaton for the language of Part (i). *(10%)*

(iii) The characters `/*` introduce a C comment, which terminates with the characters `*/`. Comments do not nest, so this string is not legal:

```
/* Not /* a */ comment.*/
```

Give a nondeterministic finite automaton that accepts the set of strings that are valid comments according to this definition. *(10%)*

Question 2 *[30%]*

(i) Consider the following simplified grammar for the Tiny programming language.

$\langle\text{program}\rangle \rightarrow \langle\text{stmtseq}\rangle$
$\langle\text{stmtseq}\rangle \rightarrow \langle\text{statement}\rangle\ \langle\text{stmtseq2}\rangle$
$\langle\text{stmtseq2}\rangle \rightarrow\ ;\ \langle\text{stmtseq}\rangle \mid \epsilon$
$\langle\text{statement}\rangle \rightarrow \langle\text{ifstmt}\rangle \mid \langle\text{repeatstmt}\rangle \mid \langle\text{assignstmt}\rangle$
$\langle\text{ifstmt}\rangle \rightarrow \textbf{if}\ \langle\text{exp}\rangle\ \textbf{then}\ \langle\text{stmtseq}\rangle\ \textbf{end}$
$\langle\text{repeatstmt}\rangle \rightarrow \textbf{repeat}\ \langle\text{stmtseq}\rangle\ \textbf{until}\ \langle\text{exp}\rangle$
$\langle\text{assignstmt}\rangle \rightarrow \textbf{id} := \langle\text{exp}\rangle$
$\langle\text{exp}\rangle \rightarrow \textbf{num}$

Give a complete parse tree for the following Tiny program.

x := 17;
if 1 then y := 23 end;
z := 29

(7%)

(ii) Give a context-free grammar for the language consisting of strings of the form 0^n1^n over the alphabet $\{0,1\}$ i.e strings containing some number of zeros followed by the same number of ones. *(3%)*

(iii) Give a context-free grammar for the language consisting of strings over the alphabet $\{0,1\}$ that contain precisely the same number of zeros and ones. *(14%)*

(iv) Draw a complete parse tree for each of the following strings: (a) 000111, (b) 010101 and (c) 00111100. *(6%)*

Question 3 *[30 %]*

(i) Compute the FIRST and FOLLOW sets for the nonterminals of the Tiny grammar given above. *(14 marks)*

(ii) Complete the parse table for this language for the stack-based, table-driven LL(1) parsing algorithm. *(10 marks)*

(iii) Write a succinct description of the stack-based, table-driven LL(1) parsing algorithm. *(6%)*

Question 4 *[10 %]*

(i) Suppose we wish to augment the Tiny language with a simple for-loop with the syntax shown below.

```
for v := s to f do
    begin
    loop body
    end
```

```
total := 0;
for i := 1 to 100 do
    begin
    total := total + i
    end
```

In the template shown above left, v is an integer variable (the loop variable), s and f are integer expressions specifying the first and last values for the loop variable and "loop body" is a placeholder for any sequence of statements bracketed by the keywords **begin** and **end**. The intended interpretation is that the loop body should be executed with variable v taking on each value in the sequence from s to f in turn inclusive.

Show how the grammar of Question 2 may be modified to incorporate this feature. *(5%)*

(ii) Give a general template for the translation of any for loop into three-address code. Use the example given above right to illustrate the approach. *(5%)*