Question 1 *[30%]*

(i) Give both a regular expression and a deterministic finite automaton that captures all alphanumeric strings (*i.e.* consisting of letters and digits) that contain the string "august" as a substring. (You may make use of the following shorthand: $\cdot$ (dot) for "any symbol" and $[\wedge x]$ for "any symbol other than x".) *(10%)*

(ii) Draw a deterministic finite automaton that captures the strings over the alphabet $\{a, b, c\}$ in the following set: Between each consecutive pair of as there is at least one b and and least one c. *(10%)*

(iii) Give a regular expression that captures the set of binary strings such that every consecutive block of five symbols contains at most three zeros. *(10%)*

Question 2 *[25%]*

(i) Consider the following simple grammar for the programming language PL0.

$\langle prog \rangle \rightarrow \langle list \rangle$
$\langle list \rangle \rightarrow \langle assignment \rangle$; $\langle list \rangle \mid \epsilon$
$\langle assignment \rangle \rightarrow$ **id** := $\langle expr \rangle$
$\langle expr \rangle \rightarrow \langle expr \rangle + \langle term \rangle \mid \langle expr \rangle - \langle term \rangle \mid \langle term \rangle$
$\langle term \rangle \rightarrow \langle term \rangle * \langle factor \rangle \mid \langle term \rangle$ **div** $\langle factor \rangle \mid \langle term \rangle$ **mod** $\langle factor \rangle \mid \langle factor \rangle$
$\langle factor \rangle \rightarrow (\langle expr \rangle) \mid$ **id** $\mid$ **num**

Give a complete parse tree for the following program.

```
x := ( 17 );
y := x + 1;
z := y;
```

(10%)

(ii) Modify the grammar so that semicolons act as statement separators rather than statement terminators (i.e. so that a three-statement program would appear as follows

```
x := 17;
y := x + 1;
z := y
```

i.e. without a semicolon after the last statement.) *(5%)*

(iii) Modify the grammar of Part (i) to incorporate a simple while-loop construct with the following syntax:

while <expr> **do**
  begin
  <list>
  end

(10%)

Question 3 *[30 %]*

(i) The following grammar is intended to capture the syntax of simple context-free grammars. The symbols $\Rightarrow$, $\bullet$, N , $\|$ and **T** are all terminals. State which nonterminals are nullable.

$$\langle G \rangle \rightarrow \langle P \rangle \langle G' \rangle$$
$$\langle G' \rangle \rightarrow \langle P \rangle \langle G' \rangle \mid \epsilon$$
$$\langle P \rangle \rightarrow \langle L \rangle \Rightarrow \langle R \rangle \bullet$$
$$\langle L \rangle \rightarrow \mathrm{N}$$
$$\langle R \rangle \rightarrow \langle A \rangle \langle A' \rangle$$
$$\langle A \rangle \rightarrow \langle S \rangle \langle S' \rangle$$
$$\langle A' \rangle \rightarrow \| \langle A \rangle \langle A' \rangle \mid \epsilon$$
$$\langle S \rangle \rightarrow \mathrm{N} \mid \mathbf{T}$$
$$\langle S' \rangle \rightarrow \langle S \rangle \langle S' \rangle \mid \epsilon$$

(5%)

(ii) Analyze the grammar to calculate the FIRST sets for each of the nonterminals in the grammar. *(10%)*

(iii) Give a succinct description of a recursive descent parser this grammar. Give pseudocode for the parsing methods for nonterminals $\langle G \rangle$, $\langle G' \rangle$ and $\langle P \rangle$. Give a crisp summary (in words) of the behaviour of the remaining parsing methods. Summarise carefully any assumptions regarding the behaviour of the scanner or any ancillary methods you rely upon. *(15%)*

Question 4 *[15 %]*

(i) Consider the code fragment based on the PL0 grammar introduced in Question 2 as augmented with the while loop of Part (iii) of that question.

```
sum := 0;
k := 1;
while k <= 100 do
    begin
    sum := sum + k;
    end;
```

Give a translation of the while loop shown into three-address code (TAC), showing clearly the correspondence between the PL0 code and the TAC. *(10%)*

(ii) Write a brief note on the relationship between the concepts of regular expression and finite automata, specifically on the expressive power of both to capture pattern in strings. *(5%)*