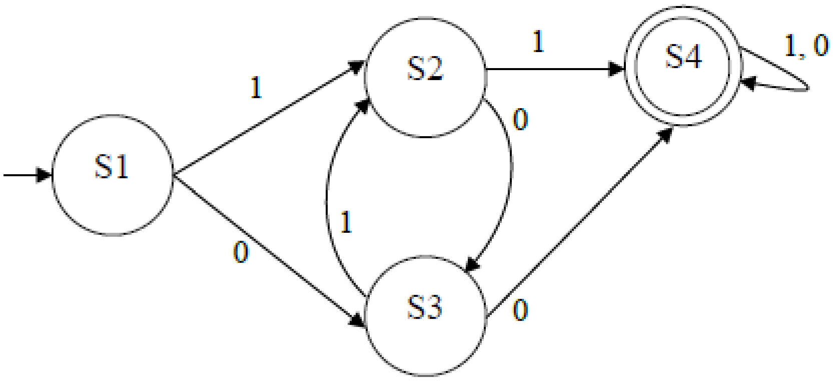
1. A finite state machine (FSM) can be used to define a language: a string is allowed in a language if it is accepted by the FSM that represents the rules of the language.
Figure 1 shows the state transition diagram for an FSM.

Figure 1



An FSM can be represented as a state transition diagram or as a state transition table. The table below is an incomplete state transition table for **Figure 1** .

(a) Complete the table.

Original state	**Input**	**New state**
S3		
S3		

(1)

(b) Any language that can be defined using an FSM can also be defined using a regular expression.

The FSM in **Figure 1** defines the language that allows all strings containing at least, either two consecutive 1s or two consecutive 0s.

The strings 0110, 00 and 01011 are all accepted by the FSM and so are valid strings in the language.

The strings 1010 and 01 are not accepted by the FSM and so are not valid strings in the language.

Write a regular expression that is equivalent to the FSM shown in **Figure 1**.

(3)

(c) Backus-Naur Form (BNF) can be used to define the rules of a language.

Figure 2 shows an attempt to write a set of BNF production rules to define a language of full names.

Figure 2

Note: underscores (_) have been used to denote spaces.
Note: rule numbers have been included but are not part of the BNF rules.

Rule number

```
1          <fullname> ::= <title>_<name>_<endtitle> |
                          <name> |
                          <title>_<name> |
                          <name>_<endtitle>

2          <title> ::= MRS | MS | MISS | MR | DR | SIR

3          <endtitle> ::= ESQUIRE | OBE | CBE

           <name> ::= <word> |
4                     <name>_<word>

5          <word> ::= <char><word>

6          <char> ::= A | B | C | D | E | F | G | H | I |
                      J | K | L | M | N | O | P | Q | R |
                      S | T | U | V | W | X | Y | Z
```

BNF can be used to define languages that are not possible to define using regular expressions. The language defined in **Figure 2** could not have been defined using regular expressions.

Complete the table below by writing either a **'Y'** for **Yes** or **'N'** for **No** in each row.

Rule number (given in Figure 2)	**Could be defined using a regular expression**
1	
2	
3	
4	
5	
6	

(1)

(d) There is an error in rule 5 in **Figure 2** which means that no names are defined by the language.

Explain what is wrong with the production rule and rewrite the production rule so that the language does define some names – the names 'BEN D JONES', 'JO GOLOMBEK' and 'ALULIM' should all be defined.

____

____

____

____

(2)

(Total 7 marks)

2. The Backus-Naur Form (BNF) production rules below define the syntax of a number of programming language constructs.

```
<forloop>    ::=  FOR <variable>  = <integer>  TO  <integer>
<variable>   ::=  <letter> | <letter> <string>
<string>     ::=  <character> | <character> <string>
<integer>    ::=  <digit> | <digit> <integer>
<character>  ::=  <digit> | <letter>
<digit>      ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

A `<letter>` is any alphabetic character from `a` to `z` or `A` to `Z`.

(a) The table below contains a list of variable names. Place a tick in each row if the stated variable name is a valid `<variable>` for the production rules above.

You may tick **more than one** box.

`<variable>`	**Valid? (✓ any number of rows)**
`a`	
`money_paid`	
`taxrate2`	
`2ndPlayerName`	

(1)

(b) The production rule for an `<integer>` is recursive.

Explain why recursion has been used in this production rule.

___

___

___

(1)

(c) Here is an example of a valid `<forloop>` :

```
FOR count = 1 TO 10
```

The BNF production rules above can be used to check whether or not a `<forloop>` is syntactically correct.

However, it is possible that a programming language statement that is a syntactically correct `<forloop>` may still produce an error when the program that it is part of is compiled.

Describe **one** example of why a syntactically correct `<forloop>` may still produce an error when a program is compiled.

___

___

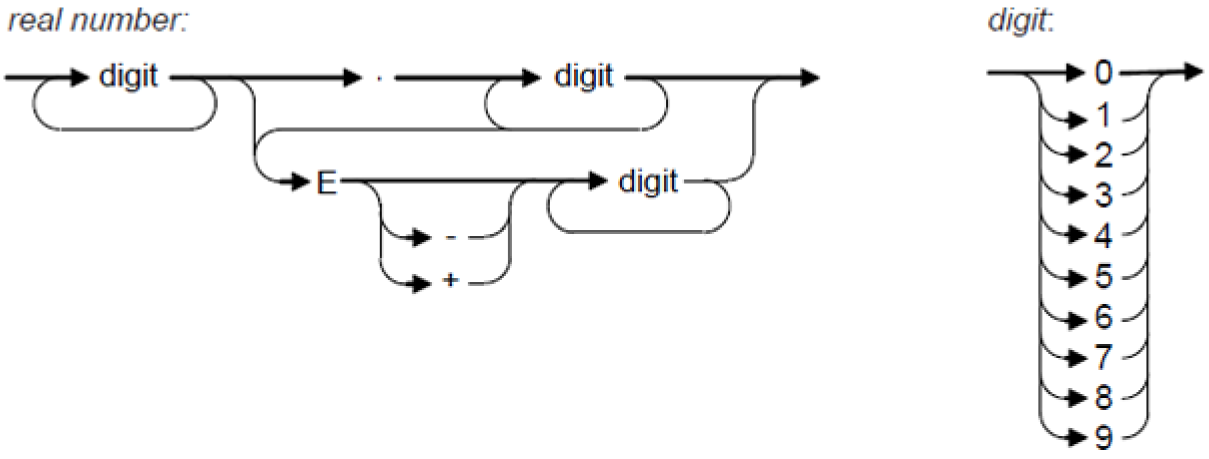
___

___

(1)

(Total 3 marks)

3. In a particular programming language, the correct syntax for a real number is defined by the syntax diagrams in the diagram below.



(a) Write **Yes** or **No** in the spaces in the empty column of the table below to identify whether or not the numbers listed in the table are valid real numbers which conform to the correct syntax for this language.

Real number	Valid? (Yes / No)
203.412	
-12.87	
12.43E-12	

(3)

(b) In the same language:

A *digit* is defined as any single numeric symbol from this list: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
A *whole number* is defined as a sequence of one or more *digits*.
An *integer* is defined as a *whole number* or a + or a – symbol followed by a *whole number*.

Write Backus-Naur Form (BNF) production rules for *digit*, *whole number* and *integer*.

____

____

____

____

(3)
(Total 6 marks)