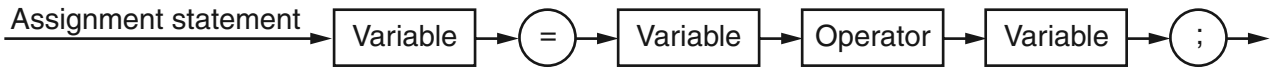
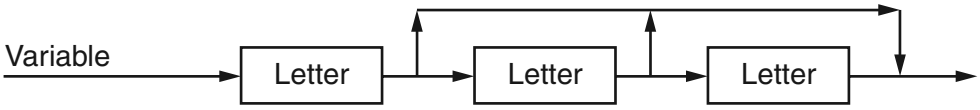
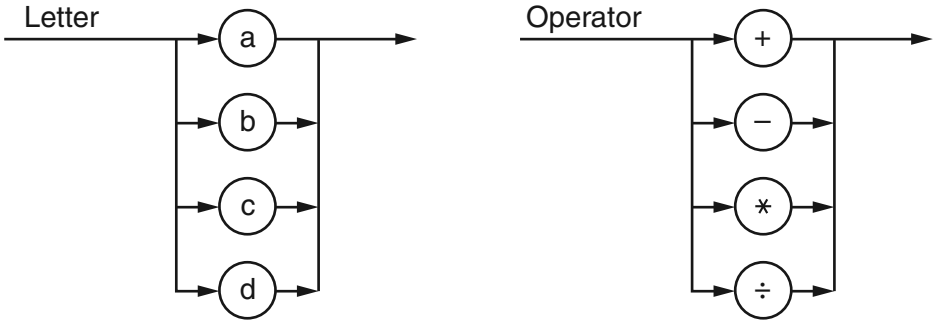
QUESTION 1.

1 The following syntax diagrams, for a particular programming language, show the

- an assignment statement
- a variable
- a letter
- an operator







(a) The following assignment statements are invalid.

Give the reason in each case.

(i) `a = b + c`

Reason .....

.....[1]

(ii) `a = b - 2;`

Reason .....

.....[1]

(iii) `a = dd * cce;`

Reason .....

.....[1]

(b) Write the Backus-Naur Form (BNF) for the syntax diagrams shown on the op

<assignmentstatement> ::=

.....

<variable> ::=

.....

<letter> ::=

.....

<operator> ::=

.....[6]

(c) Rewrite the BNF rule for a variable so that it can be any number of letters.

<variable> ::=

.....[2]

(d) Programmers working for a software development company use both interpreters and compilers.

(i) The programmers prefer to debug their programs using an interpreter.

Give **one** possible reason why.

.....

.....[1]

(ii) The company sells compiled versions of its programs.

Give a reason why this helps to protect the security of the source code.

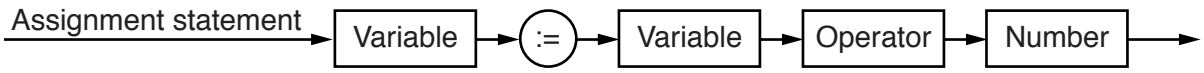
.....

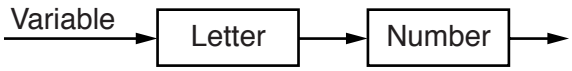
.....[1]

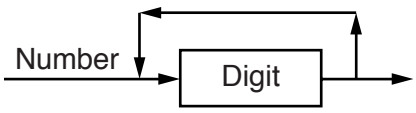
QUESTION 2.

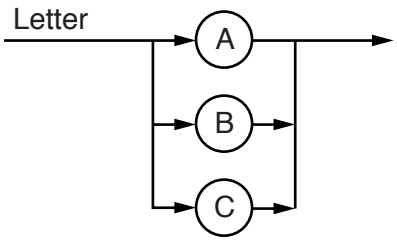
1 The following syntax diagrams, for a particular programming language, show the

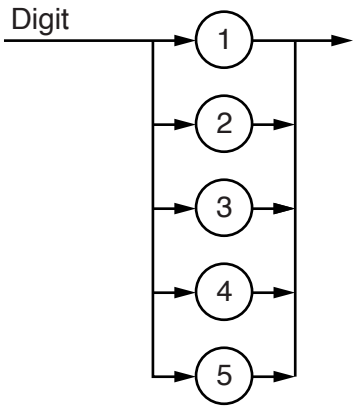
- an assignment statement
- a variable
- a number
- a letter
- a digit
- an operator

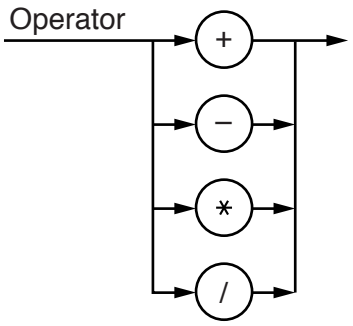












(a) The following assignment statements are invalid.

Give a reason in each case.

(i) `A2 = B3 + 123`

Reason .....

.....[1]

(ii) `B3 := B3 - 203`

Reason .....

.....[1]

(iii) `A2414 := A3 * B`

Reason .....

.....[1]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams shown on the opposite page.

<letter> has been done for you.

<assignmentstatement> ::=

.....

<variable> ::=

.....

<number> ::=

.....

<letter> ::= A | B | C

<digit> ::=

.....

<operator> ::=

.....

[6]

(c) A company develops software. It provides virtual machines for its software
company has a large number of clients who use a wide range of hardware and

(i) Explain the term virtual machine. Ensure that your answer includes the terms **h**
and **software**.

.....

.....

.....

.....[2]

(ii) Give **one** benefit to the company of using virtual machines.

.....

.....[1]

(iii) Give **one** drawback to the company of using virtual machines.

.....

.....[1]

QUESTION 3.

2 A compiler uses a keyword table and a symbol table. Part of the keyword table is

- Tokens for keywords are shown in hexadecimal.
- All the keyword tokens are in the range 00 – 5F.

Keyword	Token
←	01
+	02
=	03
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
FOR	4E
STEP	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following piece of code:

```
Counter ← 1.5
INPUT Num1
  // Check values
IF Counter = Num1
   THEN
      Num1 ← Num1 + 5.0
ENDIF
```

(a) Complete the symbol table below to show its contents after the lexical analysis stage.

Symbol	Token	
	Value	Type
Counter	60	Variable
1.5	61	Constant

[3]

(b) Each cell below represents one byte of the output from the lexical analysis st

Using the keyword table and your answer to **part (a)** complete the output from analysis.

60	01														

[2]

(c) This line of code is to be compiled:

```
A ← B + C + D
```

After the syntax analysis stage, the compiler generates object code. The equivalent code, in assembly language, is shown below:

```
LDD 234     //loads value B
ADD 235     //adds value C
STO 567     //stores result in temporary location
LDD 567     //loads value from temporary location
ADD 236     //adds value D
STO 233     //stores result in A
```

(i) Name the final stage in the compilation process that follows this code generation stage.

.....[1]

(ii) Rewrite the equivalent code given above to show the effect of it being processed through this final stage.

.....

.....

.....

.....

.....

.....[2]

(iii) State **two** benefits of the compilation process performing this final stage.

Benefit 1 .....

.....

Benefit 2 .....

.....[2]

QUESTION 4.

2 In this question, you are shown pseudocode in place of a real high-level langu
uses a keyword table and a symbol table. Part of the keyword table is shown below.

- Tokens for keywords are shown in hexadecimal.
- All the keyword tokens are in the range 00 to 5F.

Keyword	Token
←	01
+	02
=	03
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
FOR	4E
STEP	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following piece of code:

```
Start ← 0.1
// Output values in loop
FOR Counter ← Start TO 10
    OUTPUT Counter + Start
ENDFOR
```

(a) Complete the symbol table below to show its contents after the lexical analysis stage.

Symbol	Token	
	Value	Type
Start	60	Variable
0.1	61	Constant

[3]

(b) Each cell below represents one byte of the output from the lexical analysis st

Using the keyword table and your answer to **part (a)** complete the output from analysis.

60	01												

[2]

(c) The compilation process has a number of stages. The output of the lexical analysis stage forms the input to the next stage.

(i) Name this stage.

.....[1]

(ii) State **two** tasks that occur at this stage.

.....

.....

.....

.....[2]

(d) The final stage of compilation is optimisation. There are a number of reasons for performing optimisation. One reason is to produce code that minimises the amount of memory used.

(i) State another reason for the optimisation of code.

.....[1]

(ii) What could a compiler do to optimise the following expression?

```
A ← B + 2 * 6
```

.....

.....

.....[1]

(iii) These lines of code are to be compiled:

```
X ← A + B
Y ← A + B + C
```

Following the syntax analysis stage, object code is generated. The equivalent code, assembly language, is shown below:

```
LDD 436     //loads value A
ADD 437     //adds value B
STO 612     //stores result in X
LDD 436     //loads value A
ADD 437     //adds value B
ADD 438     //adds value C
STO 613     //stores result in Y
```

(iv) Rewrite the equivalent code, given above, following optimisation.

.....

.....

.....

.....

.....

.....[3]

QUESTION 5.

2 There are four stages in the compilation of a program written in a high-level langu

(a) Four statements and four compilation stages are shown below.

Draw a line to link each statement to the correct compilation stage.

Statement	Compilation stage
This stage removes any comments in the program source code.	Lexical analysis
This stage could be ignored.	Syntax analysis
This stage checks the grammar of the program source code.	Code generation
This stage produces a tokenised version of the program source code.	Optimisation

[4]

(b) Write the Reverse Polish Notation (RPN) for the following expressions.

(i) `(A + B) * (C - D)`

..... [2]

(ii) `-A / B * 4 / (C - D)`

..... [3]

(c) An interpreter is executing a program. The program uses the variables `w`, `x`, `y`

The program contains an expression written in infix form. The interpreter conve
expression to RPN. The RPN expression is:

```
x w z + y - *
```

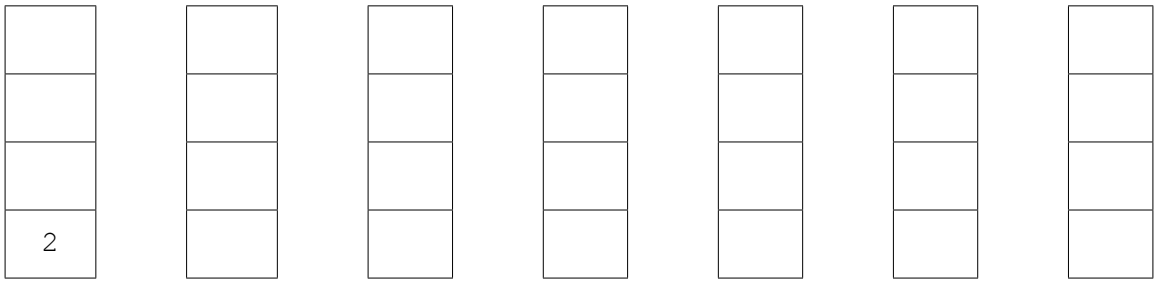
The interpreter evaluates this RPN expression using a stack.

The current values of the variables are:

```
w = 1   x = 2   y = 3   z = 4
```

(i) Show the changing contents of the stack as the interpreter evaluates the expression.

The first entry on the stack has been done for you.



[4]

(ii) Convert back to its original infix form, the RPN expression:

```
x w z + y - *
```

.....

..... [2]

(iii) Explain **one** advantage of using RPN for the evaluation of an expression.

.....

.....

.....

..... [2]

QUESTION 6.

2 There are four stages in the compilation of a program written in a high-level langu

(a) Four statements and four compilation stages are shown below.

Draw a line to link each statement to the correct compilation stage.

Statement	Compilation stage
This stage can improve the time taken to execute the statement: `x = y + 0`	Lexical analysis
This stage produces object code.	Syntax analysis
This stage makes use of tree data structures.	Code generation
This stage enters symbols in the symbol table.	Optimisation

[4]

(b) Write the Reverse Polish Notation (RPN) for the following expression.

```
P + Q - R / S
```

..... [2]

(c) An interpreter is executing a program. The program uses the variables `a`, `b`, `c`

The program contains an expression written in infix form. The interpreter conver
expression to RPN. The RPN expression is:

```
b a * c d a + + -
```

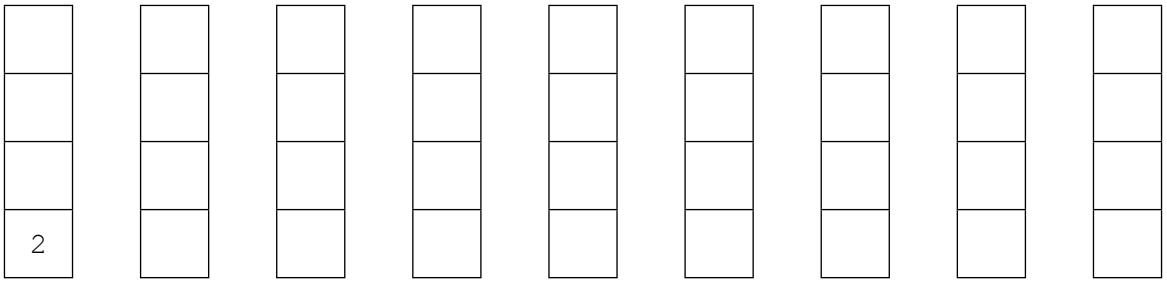
The interpreter evaluates this RPN expression using a stack.

The current values of the variables are:

```
a = 2   b = 2   c = 1   d = 3
```

(i) Show the changing contents of the stack as the interpreter evaluates the expression.

The first entry on the stack has been done for you.



[4]

(ii) Convert back to its original infix form, the RPN expression:

```
b a * c d a + + -
```

.....

..... [2]

(iii) One advantage of using RPN is that the evaluation of an expression does not require rules of precedence.

Explain this statement.

.....

.....

.....

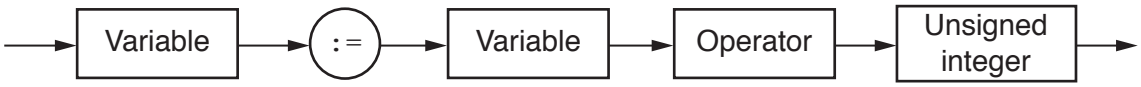
..... [2]

QUESTION 7.

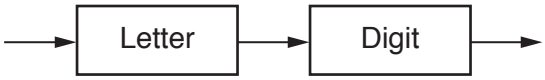
3 The following syntax diagrams for a particular programming language show the s

- an assignment statement
- a variable
- an unsigned integer
- a letter
- an operator
- a digit.

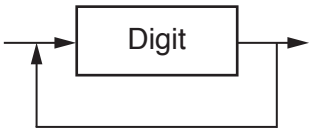
Assignment statement



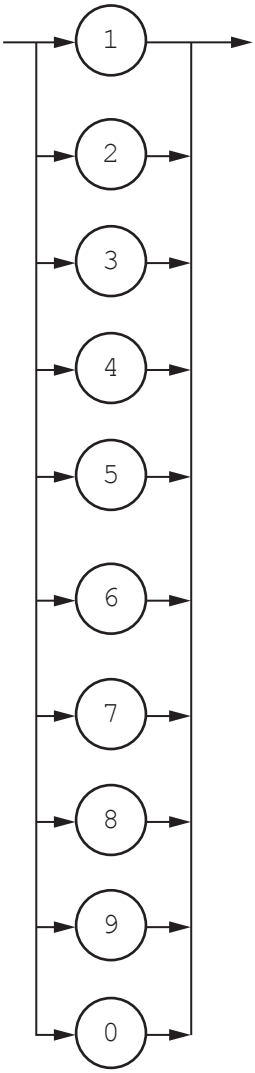
Variable



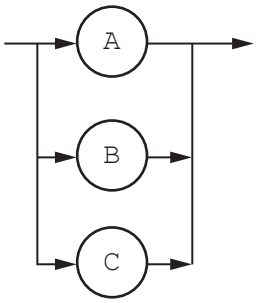
Unsigned integer



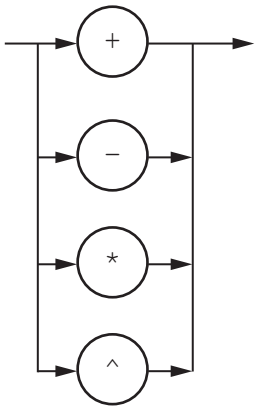
Digit



Letter



Operator



(a) The following assignment statements are invalid.

Give the reason in each case.

(i) `C2 = C3 + 123`

Reason: .....

..... [1]

(ii) `A3 := B1 - B2`

Reason: .....

..... [1]

(iii) `A32 := A2 * 7`

Reason: .....

..... [1]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams shown.

`<digit>` has been done for you.

`<assignment_statement> ::=`

.....

`<variable> ::=`

.....

`<unsigned_integer> ::=`

.....

`<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0`

`<letter> ::=`

.....

`<operator> ::=`

.....

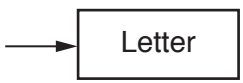
[6]

(c) The definition of <variable> is changed to allow:

- one or two letters and
- zero, one or two digits.

Draw an updated version of the syntax diagram for <variable>.

Variable



[2]

(d) The definition of <assignment_statement> is altered so that its syntax has <unsigned_integer> replaced by <real>.

A real is defined to be:

- at least one digit before a decimal point
- a decimal point
- at least one digit after a decimal point.

Give the BNF for the revised <assignment_statement> and <real>.

<assignment_statement> ::= .....

.....

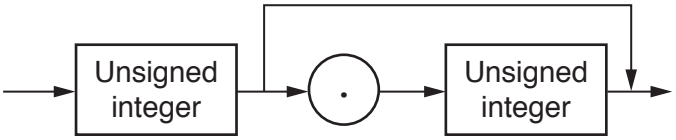
<real> ::= .....

QUESTION 8.

4 The following syntax diagrams for a particular programming language show the s

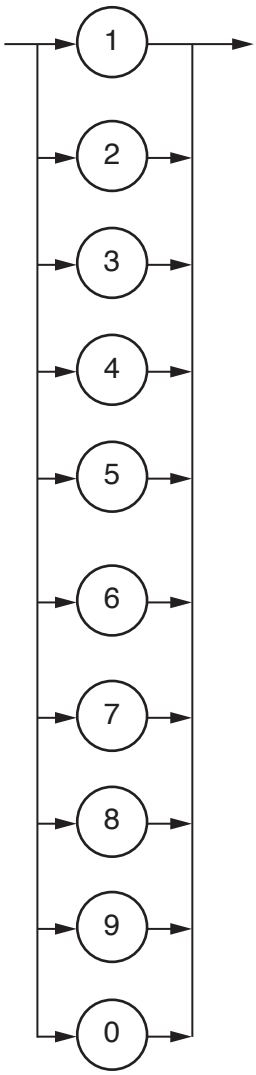
- an unsigned number
- an unsigned integer
- a digit.

Unsigned number



Unsigned integer

Digit



(a) (i) Explain why 32 is a valid unsigned integer.

.....

.....

.....

.....[2]

(ii) Explain why 32.5 is a valid unsigned number.

.....

.....

.....

.....[2]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams shown.

<unsigned_number> ::= .....

.....

<unsigned_integer> ::= .....

.....

<digit> ::= .....

.....

[5]

The format of an unsigned number is amended to include numbers with possible exponents.

If an unsigned number has an exponent, then the exponent part:

- will start with an 'E'
- be followed by an optional '+' or '–' sign
- and be completed by an unsigned integer.

Examples of unsigned numbers with exponents include: 3E2, 3E+3, 3E–32, 3.45E–2

(c) **(i)** Redraw the syntax diagram for unsigned number to include numbers that might have exponents.

[4]

(ii) Use your syntax diagram from **part (c)(i)** to write the BNF for an uns
include numbers with exponents.

<unsigned_number> ::= .....

.....

.....

.....

.....

.....

.....

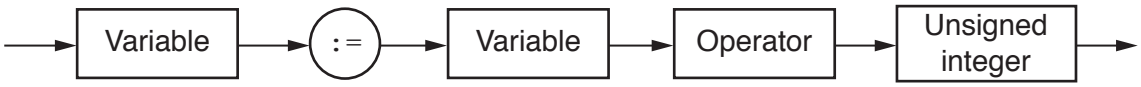
.....[4]

QUESTION 9.

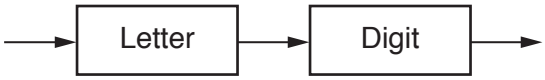
3 The following syntax diagrams for a particular programming language show the s

- an assignment statement
- a variable
- an unsigned integer
- a letter
- an operator
- a digit.

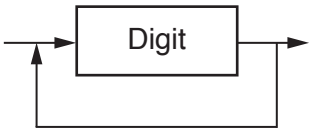
Assignment statement



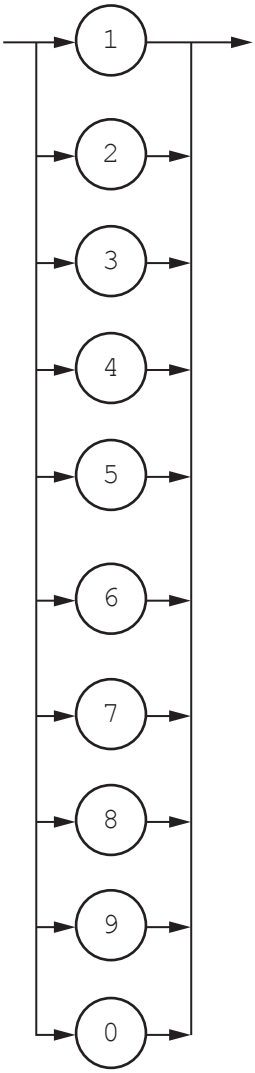
Variable



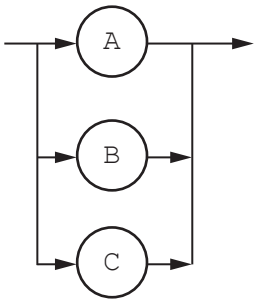
Unsigned integer



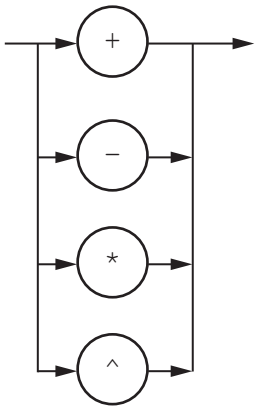
Digit



Letter



Operator



(a) The following assignment statements are invalid.

Give the reason in each case.

(i) `C2 = C3 + 123`

Reason: .....

..... [1]

(ii) `A3 := B1 - B2`

Reason: .....

..... [1]

(iii) `A32 := A2 * 7`

Reason: .....

..... [1]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams shown.

`<digit>` has been done for you.

`<assignment_statement> ::=`

.....

`<variable> ::=`

.....

`<unsigned_integer> ::=`

.....

`<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0`

`<letter> ::=`

.....

`<operator> ::=`

.....

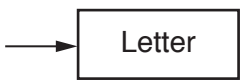
[6]

(c) The definition of <variable> is changed to allow:

- one or two letters and
- zero, one or two digits.

Draw an updated version of the syntax diagram for <variable>.

Variable



[2]

(d) The definition of <assignment_statement> is altered so that its syntax has <unsigned_integer> replaced by <real>.

A real is defined to be:

- at least one digit before a decimal point
- a decimal point
- at least one digit after a decimal point.

Give the BNF for the revised <assignment_statement> and <real>.

<assignment_statement> ::= .....

.....

<real> ::= .....

QUESTION 10.

6 The compilation process has a number of stages. The first stage is lexical analys

A compiler uses a keyword table and a symbol table. Part of the keyword table is show

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range `00` – `5F`.

Keyword	Token
←	01
*	02
=	03
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
FOR	4E
STEP	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following code.

```
Start ← 1
INPUT Number
// Output values in a loop
FOR Counter ← Start TO 12
    OUTPUT Number * Counter
ENDFOR
```

(a) Complete the symbol table to show its contents after the lexical analysis stag

Symbol	Token	
	Value	Type
`Start`	`60`	Variable
`1`	`61`	Constant

[3]

(b) The output from the lexical analysis stage is stored in the following table. Each cell stores one byte of the output.

Complete the output from the lexical analysis stage. Use the keyword table and your answer to **part (a)**.

`60`	`01`														

[2]

(c) The output of the lexical analysis stage is the input to the syntax analysis stage.

Identify **two** tasks in syntax analysis.

1 .....

.....

2 .....

.....

[2]

(d) The final stage of compilation is optimisation.

(i) Code optimisation produces code that minimises the amount of memory used.

Give **one** additional reason why code optimisation is performed.

.....

.....[1]

(ii) A student uses the compiler to compile some different code.

After the syntax analysis stage is complete, the compiler generates object co

The following lines of code are compiled.

```
X  ←  A + B
Y  ←  A + B + C
Z  ←  A + B + C + D
```

The compilation produces the following assembly language code.

```
LDD 236     //     loads value A to accumulator
ADD 237     //     adds value B to accumulator
STO 512     //     stores accumulator in X
LDD 236     //     loads value A to accumulator
ADD 237     //     adds value B to accumulator
ADD 238     //     adds value C to accumulator
STO 513     //     stores accumulator in Y
LDD 236     //     loads value A to accumulator
ADD 237     //     adds value B to accumulator
ADD 238     //     adds value C to accumulator
ADD 239     //     adds value D to accumulator
STO 514     //     stores accumulator in Z
```

Rewrite the assembly language code after it has been optimised.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

BLANK PAGE

QUESTION 11.

4 A compiler uses a keyword table and a symbol table. Part of the keyword table is

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range `00 – 5F`.

Keyword	Token
←	01
+	02
=	03
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
FOR	4E
STEP	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from `60` (hexadecimal).

Study the following code.

```
INPUT Number1
INPUT Number2
INPUT Answer
IF Answer = Number1 + Number2
   THEN
      OUTPUT 10
   ELSE
      OUTPUT 0
ENDIF
```

(a) Complete the symbol table to show its contents after the lexical analysis stage.

Symbol	Token	
	Value	Type
Number1	60	Variable
Number2	61	Variable

[3]

(b) The output from the lexical analysis stage is stored in the following table. Eac
byte of the output.

Complete the output from the lexical analysis. Use the keyword table and your a
part (a).

51	60																	

[2]

(c) A student uses the compiler to compile some different code.

After the syntax analysis is complete, the compiler generates object code.

The following line of code is compiled: `X ← A + B + C - D`

The compilation produces the following assembly language code.

```
LDD 236      // loads value A into accumulator
ADD 237      // adds value B to accumulator
ADD 238      // adds value C to accumulator
STO 540      // stores accumulator in temporary location
LDD 540      // loads value from temporary location into accumulator
SUB 239      // subtracts value D from accumulator
STO 235      // stores accumulator in X
```

(i) Identify the final stage in the compilation process that follows this code generation stage.

.....[1]

(ii) Rewrite the equivalent code following the final stage.

.....

.....

.....

.....

.....

.....

.....

.....[3]

(iii) State **two** benefits of the process that is carried out in the final stage.

Benefit 1 .....

.....

Benefit 2 .....

.....

[2]

(d) An interpreter is executing a program. The program uses the variables `a`, `b`, `c` and `d`.

The program contains an expression that is written in infix form. The interpreter converts the infix expression to RPN.

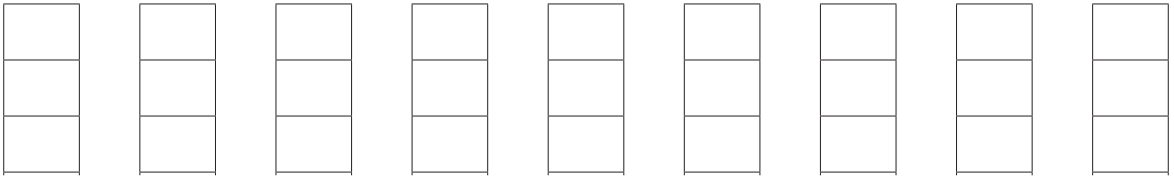
The RPN expression is: `b a c + * d + 2 -`

The interpreter evaluates this RPN expression using a stack.

The current values are: `a = 1` `b = 2` `c = 2` `d = 3`

Show the changing contents of the stack as the interpreter evaluates the expression.

The first entry on the stack has been done for you.



QUESTION 12.

4 A compiler uses a keyword table and a symbol table. Part of the keyword table is

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range 00 – 5F.

Keyword	Token
←	01
+	02
=	03
<>	04
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
REPEAT	4E
UNTIL	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following piece of pseudocode.

```
Counter ← 0
INPUT Password
REPEAT
   IF Password <> "Cambridge"
      THEN
         INPUT Password
   ENDIF
   Counter ← Counter + 1
UNTIL Password = "Cambridge"
OUTPUT Counter
```

(a) Complete the symbol table to show its contents after the lexical analysis stag

Symbol	Token	
	Value	Type
Counter	60	Variable

[3]

(b) The output from the lexical analysis stage is stored in the following table. Each cell stores one byte of the output.

Complete the output from the lexical analysis using the keyword table **and** your answer to **part (a)**.

60	01																						

[2]

(c) The following table shows assembly language instructions for a processor. general purpose register, the Accumulator (ACC).

Instruction		Explanation
Op code	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
ADD	<address>	Add the contents of the given address to the ACC
STO	<address>	Store the contents of ACC at the given address

After the syntax analysis is completed successfully, the compiler generates object code.

The following lines of high level language code are compiled.

```
X = X + Y
Z = Z + X
```

The compilation produces the assembly language code as follows:

```
LDD 236
ADD 237
STO 236
LDD 238
ADD 236
STO 238
```

(i) The final stage in the compilation process that follows this code generation stage is code optimisation.

Rewrite the equivalent code after optimisation.

.....

.....

.....

.....

.....

..... [3]

(ii) Explain why code optimisation is necessary.

.....

.....

.....

..... [2]

QUESTION 13.

4 **(a)** Describe the main steps in the evaluation of a Reverse Polish Notation (r
using a stack.

.....

.....

.....

.....

.....

.....

.....

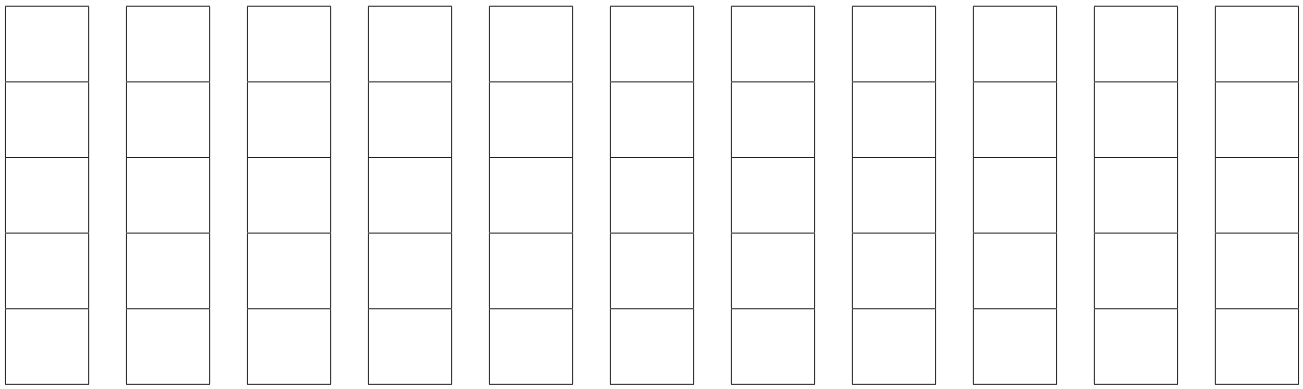
..... [4]

(b) The infix expression `8 * (5 - 2) - 30 / (2 * 3)` converts to:

`8 5 2 - * 30 2 3 * / -`

in Reverse Polish Notation (RPN).

Show the changing contents of the stack as this RPN expression is evaluated.

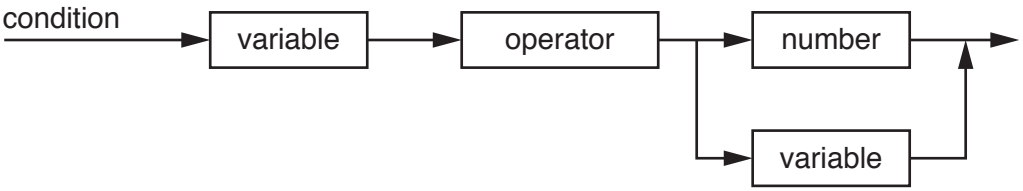


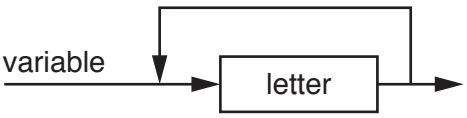
[4]

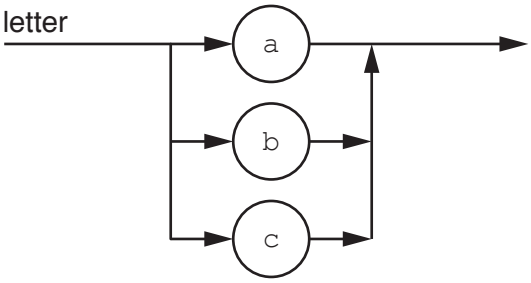
QUESTION 14.

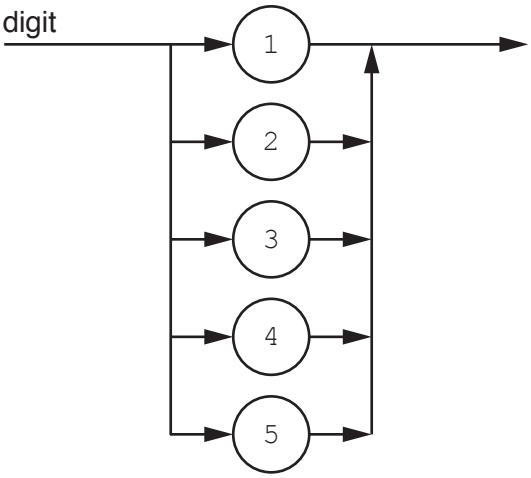
2 The following syntax diagrams for a programming language show the syntax of:

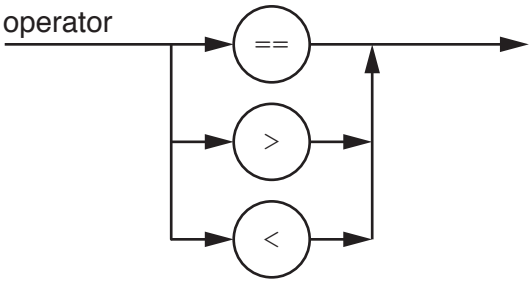
- a condition
- a variable
- a number
- a letter
- a digit
- an operator











(a) The following conditions are invalid.

Give the reason in each case.

(i) `35 > 24`

Reason .....

..... [1]

(ii) `abc := cba`

Reason .....

..... [1]

(iii) `bc < 49`

Reason .....

..... [1]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagram.

`<operator> ::=` .....

.....

`<number> ::=` .....

.....

`<variable> ::=` .....

.....

`<condition> ::=` .....

.....

[6]

QUESTION 15.

7 The following are the first few lines of a source code program written in a high-lev
source code program is to be translated by the language compiler.

```
// program written on 15 June 2019

DECLARE IsFound : Boolean;
DECLARE NoOfChildren : Integer;
DECLARE Count : Integer;
Constant TaxRate = 15;

// start of main program
For Count = 1 to 50
…
…
…
```

(a) During the lexical analysis stage, the compiler will use a keyword table and a symbol table.

(i) Identify **two** types of data in the keyword table.

Type 1 .....

Type 2 .....

[2]

(ii) Identify **two** types of data in the symbol table.

Type 1 .....

Type 2 .....

[2]

(iii) Explain how the contents of the keyword and symbol tables are used to translate the source code program.

.....

.....

.....

..... [2]

(iv) State **one** additional task completed at the lexical analysis stage that does not involve the use of a keyword or a symbol table.

.....

..... [1]

(b) The final stage of compilation can be code optimisation.

Explain why code is optimised.

.....

.....

.....

..... [2]