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

– Tokens for keywords are shown in hexadecimal.

Keyword	Token
←	01
+	02
=	03

– All the keyword tokens are in the range 00 to 5F.

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) Copy and complete this symbol table to show its contents after the lexical analysis stage.

[3]

	Token	
Symbol	**Value**	**Type**
Start	60	Variable
0.1	61	Constant

b) Each cell below represents one byte of the output from the lexical analysis stage. Using the keyword table and your answer to part a), copy and complete the output from the lexical analysis.

[2]

60	01												

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?

[1]

A ← B + 2 * 6

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, in 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
```

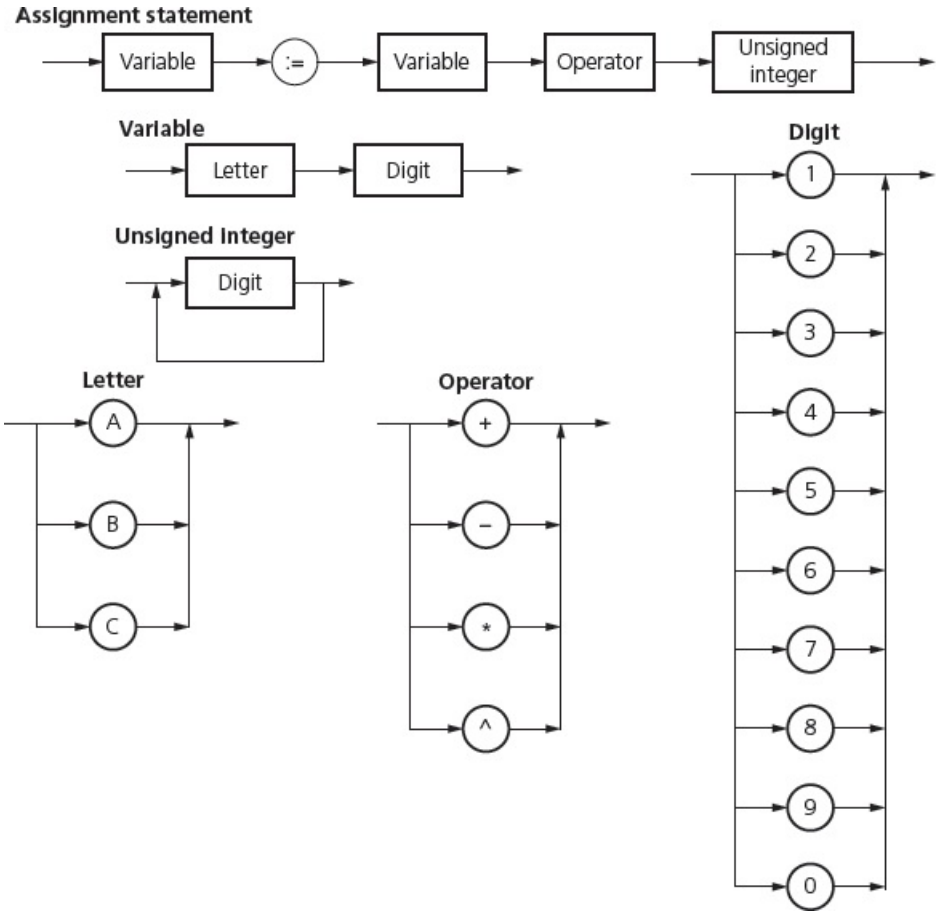
iv) Rewrite the equivalent code, given above, following optimisation.

[3]

Cambridge International AS & A Level Computer Science 9608
Paper 32 Q2 November 2015

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

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



a) The following assignment statements are invalid.
Give the reason in each case.

i) C2 = C3 + 123

[1]

ii) A3 := B1 – B2

[1]

iii) A32 := A2 * 7

[1]

b) Copy and complete the Backus-Naur Form (BNF) for the syntax diagrams shown.
<digit> has been done for you.

[6]

<assignment_statement> ::=

<variable> ::=

<unsigned_integer> ::=

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

<letter> ::=

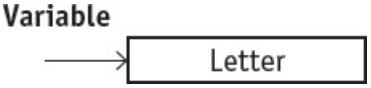
<operator> ::=

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>.



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>.

[2]

<assignment_statement> ::=

<real> ::=

Cambridge International AS & A Level Computer Science 9608
Paper 31 Q3 November 2017

9 There are four stages in the compilation of a program written in a high-level language.

a) Four statements and four compilation stages are shown below. Copy the diagram below and connect each statement to the correct compilation stage.

[4]

Statement	Compilation stage
This stage can improve the time taken to execute `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

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

[2]

P + Q – R / S

c) An interpreter is executing a program.

The program uses the variables a, b, c and d.

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

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

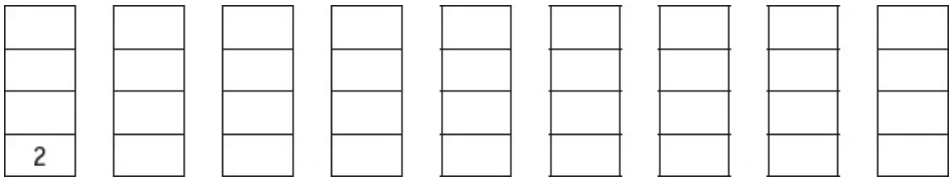
The interpreter evaluates this RPN expression using a stack.

The current values of the variables are:

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

i) Copy the diagram below and 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:

[2]

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

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

Explain this statement. [2]

Cambridge International AS & A Level Computer Science 9608
Paper 32 Q2 November 2016