Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER				

0123456789

COMPUTER SCIENCE **9618/03**

Paper 3 Advanced Theory **For examination from 2021**

SPECIMEN PAPER **1 hour 30 minutes**

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

INFORMATION

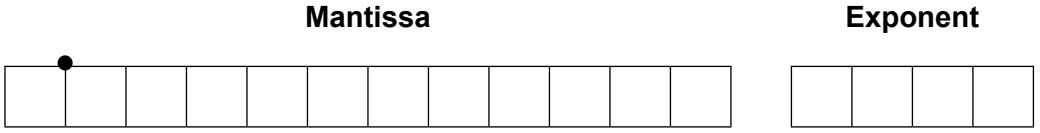
- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **16** pages. Blank pages are indicated.

1 In a particular computer system, real numbers are stored using floating-point representation with:

- 12 bits for the mantissa
- 4 bits for the exponent
- two's complement form for both mantissa and exponent.

(a) Calculate the normalised floating-point representation of +4.5 in this system. Show your working.



Working .....

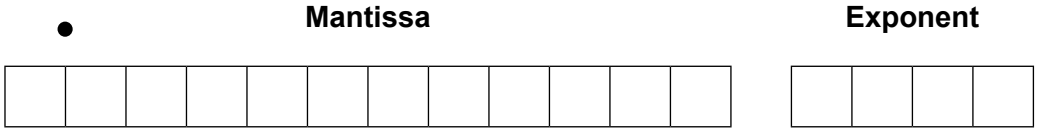
.....

.....

.....

..... [3]

(b) Calculate the normalised floating-point representation of −4.5 in this system. Show your working.



Working .....

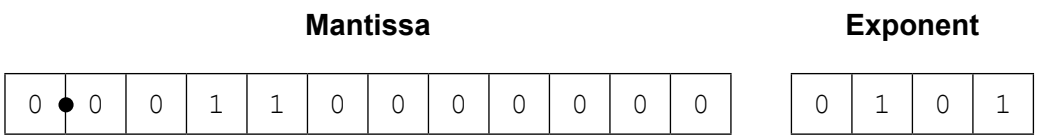
.....

.....

.....

..... [3]

(c) Calculate the denary value for the following binary floating-point number. Show your working.



Working .....

.....

.....

.....

.....

Answer .....

[3]

(d) (i) State whether the floating-point number given in **part (c)** is normalised or not normalised.

..... [1]

(ii) Justify your answer given in **part (d)(i)**.

.....

..... [1]

(e) The system changes so that it now allocates eight bits to both the mantissa and the exponent.

Explain **two** effects this has on the numbers that can be represented.

1 .....

.....

.....

2 .....

.....

.....

[4]

2 The TCP/IP protocol suite can be viewed as a stack with **four** layers.

(a) Complete the stack by inserting the names of the **three** missing layers.

Application layer

[3]

(b) BitTorrent is a protocol used at the Application layer for the exchange of data.

(i) State the network model used with this protocol.

..... [1]

(ii) State the use of BitTorrent.

..... [1]

(iii) Explain how applications use BitTorrent to exchange data.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) State **two** other protocols that are used at the Application layer for the exchange of data.

For each protocol, give a different example of an appropriate exchange of data.

Protocol 1 .....

Example .....

.....

Protocol 2 .....

Example .....

.....

[4]

3 **(a)** Complete the Boolean expession that corresponds to the following truth table.

INPUT			OUTPUT
A	**B**	**C**	**X**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

X = $\overline{A}$. B . C ..... [2]

The part to the right of the equals sign is known as the sum-of-products.

(b) **(i)** Complete the Karnaugh map (K-map) for the truth table given in **part (a)**.

		AB			
		00	**01**	**11**	**10**
C	**0**				
	1				

[1]

The K-map can be used to simplify the function in **part (a)**.

(ii) Draw loop(s) around appropriate groups of 1s to produce an optimal sum-of-products. [2]

(iii) Using your answer to **part (b)(ii)**, write the simplified sum-of-products Boolean expression.

X = ... [2]

4 A student writes a program in a high-level programming language. A compiler translates the program into machine code.

(a) The compilation process has a number of stages.

The output of the lexical analysis stage forms the input to the next stage.

(i) Identify this stage.

..... [1]

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

1 .....

.....

2 .....

.....

[2]

(b) The program uses pseudocode in place of a real high-level language.

There are a number of reasons for performing optimisation. One reason is to produce code that minimises the amount of memory used.

State another reason for the optimisation of code.

..... [1]

(c) The following statement assigns an expression to the variable `A`.

Suggest what a compiler could do to optimise the following expression.

```
A ← B + 2 * 6
```

.....

.....

..... [1]

(d) 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:

```
01  LDD 436 //loads value A
02  ADD 437 //adds value B
03  STO 612 //stores result in X
04  LDD 436 //loads value A
05  ADD 437 //adds value B
06  ADD 438 //adds value C
07  STO 613 //stores result in Y
```

Suggest what a compiler could do to optimise this code.

.....

.....

.....

.....

.....

..... [3]

5 Ed wants to send a message securely. Before sending the message, software encrypts it using a symmetric key

(a) (i) Describe what is meant by **symmetric key encryption**.

.....

.....

.....

.....

.....

..... [2]

(ii) State **two** drawbacks of using symmetric key encryption.

.....

.....

.....

.....

.....

..... [2]

(b) The symmetric key is to be exchanged before the message is sent.
To exchange the key securely, the use of quantum cryptography is being considered.

State **two** possible benefits of using quantum cryptography.

.....

.....

.....

.....

.....

..... [2]

6 **(a)** Artificial Intelligence (AI) can be aided by the use of different techniques.

Draw a line from each technique to the correct description.

Technique	Description
	A structure used to model relationships between objects.
Artificial Neural Network	A computer system modelled on a brain.
A* Algorithm	A computer program that improves its performance at certain tasks with experience.
Graph	An abstract data type with a hierarchical structure.
Machine Learning	A computer method used to find the optimal path between two mapped locations.

[4]

(b) Describe **two** categories of machine learning.

1 .....

.....

.....

.....

2 .....

.....

.....

.....

[4]

7 An ordered binary tree Abstract Data Type (ADT) has these associated operations:

- create tree
- add new item to tree
- traverse tree

A student is designing a program that will implement a binary tree ADT as a linked list of **ten** nodes.

Each node consists of data, a left pointer and a right pointer.

A program is to be written to implement the tree ADT. The variables and procedures to be used are listed below:

Identifier	Data type	Description
`Node`	`RECORD`	Data structure to store node data and associated pointers.
`LeftPointer`	`INTEGER`	Stores index of start of left subtree.
`RightPointer`	`INTEGER`	Stores index of start of right subtree.
`Data`	`STRING`	Data item stored in node.
`Tree`	`ARRAY`	Array to store nodes.
`NewDataItem`	`STRING`	Stores data to be added.
`FreePointer`	`INTEGER`	Stores index of start of free list.
`RootPointer`	`INTEGER`	Stores index of root node.
`NewNodePointer`	`INTEGER`	Stores index of node to be added.
`CreateTree()`		Procedure initialises the root pointer and free pointer and links all nodes together into the free list.
`AddToTree()`		Procedure to add a new data item in the correct position in the binary tree.
`FindInsertionPoint()`		Procedure that finds the node where a new node is to be added. Procedure takes the parameter `NewDataItem` and returns two parameters: `Index`, whose value is the index of the node where the new node is to be added `Direction`, whose value is the direction of the pointer (“`Left`” or “`Right`”).

These pseudocode declarations and this procedure can be used to create an empty tree with ten nodes.

```
TYPE Node
    DECLARE LeftPointer : INTEGER
    DECLARE RightPointer: INTEGER
    DECLARE Data : STRING
ENDTYPE
DECLARE Tree  : ARRAY[0 : 9] OF Node
DECLARE FreePointer : INTEGER
DECLARE RootPointer : INTEGER

PROCEDURE CreateTree()
    DECLARE Index : INTEGER
    RootPointer ← -1
    FreePointer ← 0
    FOR Index ← 0 TO 9  // link nodes
        Tree[Index].LeftPointer ← Index + 1
        Tree[Index].RightPointer ← -1
    NEXT
    Tree[9].LeftPointer ← -1
ENDPROCEDURE
```

(a) Complete the pseudocode to add a data item to the tree.

```
PROCEDURE AddToTree(BYVALUE NewDataItem : STRING)
// if no free node report an error

    IF FreePointer .....
        THEN
            OUTPUT "No free space left"
        ELSE
            // add new data item to first node in the free list
            NewNodePointer ← FreePointer

            .....
            // adjust free pointer

            FreePointer ← .....
            // clear left pointer

            Tree[NewNodePointer].LeftPointer ← .....
            // is tree currently empty?

            IF .....
                THEN // make new node the root node

                    .....
                ELSE // find position where new node is to be added
                    Index ← RootPointer
                    CALL FindInsertionPoint(NewDataItem, Index, Direction)
                    IF Direction = "Left"
                        THEN // add new node on left

                            .....
                        ELSE // add new node on right

                            .....
                    ENDIF
            ENDIF
    ENDIF
ENDPROCEDURE
```

[8]

(b) The traverse tree operation outputs the data items in alphabetical order.
This can be written as a recursive solution.

Complete the pseudocode for the recursive procedure `TraverseTree`.

```
PROCEDURE TraverseTree(BYVALUE Pointer : INTEGER)
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDPROCEDURE
```

[5]

8 The table shows assembly language instructions for a processor that has one general purpose register, the Accumulator (ACC).

Instruction		Explanation
Opcode	**Operand**	
LDM	#n	Load the denary number n to ACC
LDD	<address>	Load the contents of the given address to ACC
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
INC	<register>	Add 1 to the contents of the register
CMP	<address>	Compare the contents of ACC with the contents of <address>
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
END		Return control to the operating system

(a) State the addressing mode used by:

LDM .....

.....

LDD .....

.....
[2]

(b) Using opcodes from the table, write instructions to set the value at address 509 to the contents of address 500 added to the value 12.

.....

.....

.....

..... [3]

BLANK PAGE