Programming

Question 1

1 Carlos is writing exception handling code for his program.

(a) State what is meant by an **exception**.

.....

..... [1]

(b) Give **three** situations where an exception handling routine would be required.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(c) Describe the benefits of using exception handling in a program.

.....

.....

.....

..... [2]

Question 2

2 **(a)** Programs can be written using recursion.

Tick (✓) one or more boxes to show the features that **must** be included in a valid recursive algorithm.

Feature	Must be included
Incrementation	
General case	
Base case	
Selection case	
It calls itself	

[2]

(b) The following recursive procedure outputs every even number from the positive parameter value down to and including 2.

The procedure checks if the integer parameter is an even or an odd number. If the number is odd, the procedure converts it to an even number by subtracting 1 from it.

The function `MOD(ThisNum : INTEGER, ThisDiv : INTEGER)` returns the remainder value when `ThisNum` is divided by `ThisDiv`.

Complete the **pseudocode** for the recursive procedure.

```
PROCEDURE Count(BYVALUE ..... : INTEGER)
    IF .....(Number, 2) <> 0
        THEN
            Number ← Number - 1
    ENDIF
    OUTPUT .....
    IF Number > 0
        THEN
            .....(..... - 1)
    ENDIF
ENDPROCEDURE
```

[5]

(c) A program allows guests to input a meal option at a wedding.

Guests can choose meal option 1 or meal option 2.

The program will keep count of the numbers of each meal option chosen.

The program ends when a value other than 1 or 2 is entered. It then outputs the count of each meal option.

```
PROCEDURE MealsCount(BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)

    DECLARE MealOption : INTEGER

    DECLARE MoreMeals : BOOLEAN

    MoreMeals ← True

    WHILE MoreMeals = True

        INPUT MealOption

        IF MealOption = 1

            THEN

                MealOption1 ← MealOption1 + 1

            ELSE

                IF MealOption = 2

                    THEN
                        MealOption2 ← MealOption2 + 1

                     ELSE

                        OUTPUT MealOption1, " ", MealOption2

                        MoreMeals ← False

                 ENDIF

         ENDIF

     ENDWHILE

ENDPROCEDURE
```

The program contains a conditional loop.

Use **pseudocode** to rewrite the conditional loop as a recursive algorithm.

```
PROCEDURE MealsCount(BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)
    DECLARE MealOption : INTEGER
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
ENDPROCEDURE
```

[5]

Question 3

3 A declarative programming language is used to represent the following knowledge base.

```
01 person(jessica).

02 person(pradeep).

03 person(steffi).

04 person(johann).

05 sport(football).

06 sport(hockey).

07 sport(cricket).

08 sport(volleyball).

09 plays(johann, football).

10 plays(steffi, cricket).

11 plays(jessica, football).

12 will_not_play(pradeep, cricket).
```

These clauses have the following meanings:

Clause	Meaning
01	Jessica is a person
05	Football is a sport
09	Johann plays football
12	Pradeep refuses to play cricket

(a) Elle is a person who plays rugby but refuses to play hockey.

Write additional clauses to represent this information.

13 .....

14 .....

15 .....

16 .....

[4]

(b) Write the result returned by the goal:

```
plays(X, football).
```

X = ... [1]

(c) Y might play X, if Y is a person, X is a sport and Y does not refuse to play X.

Write this as a rule.

```
mightplay(Y , X)
```

IF .....

.....

..... [5]

Question 4

4 Object-oriented programming has several features. These include inheritance, classes, methods and properties.

(a) Describe what is meant by **inheritance**.

.....

.....

.....

..... [2]

(b) Identify **two other** features of object-oriented programming.

1 .....

2 .....

[2]

Question 5

6 A theatre company stores customer login details to allow customers to book tickets online.

A hash table stores login details for 2000 customers.

Each customer's details are stored in a record.

The declaration for `CustomerRecord` is:

```
TYPE CustomerRecord

      DECLARE UserID : STRING

      DECLARE PINNumber : INTEGER

ENDTYPE
```

A 1D array, `CustomerDetails`, is used to implement the hash table. `CustomerDetails` is a global array. The 1D array has 6000 elements.

(a) The procedure `InitialiseHashTable()` initialises the hash table. `UserID` is initialised as an empty string, and `PINNumber` initialised to `0` for all of the records.

Write **pseudocode** for the procedure `InitialiseHashTable()`.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) The function `InsertRecord()` is used to insert a new record into the hash table.

The function, `Hash()`:

- takes a `UserID` as a parameter
- performs the hashing algorithm
- returns the calculated index of the user ID within the hash table.

If the hash table is full, the function `InsertRecord()` returns `-1`. If there is space available in the hash table, the record is inserted, and it returns the position of this record in the array.

Complete the **pseudocode** for the function.

```
FUNCTION InsertRecord(NewRecord : CustomerRecord) RETURNS INTEGER
    DECLARE Count : INTEGER
    DECLARE Index : INTEGER
    Count ← 0
    Index ← Hash(.....)
    WHILE (CustomerDetails[Index].UserID <> "") ..... (Count <= 5999)
        Index ← Index + 1
        Count ← Count + 1
        IF Index > 5999
            THEN
                .....
        ENDIF
    ENDWHILE
    IF Count > 5999
       THEN
           .....
       ELSE
           CustomerDetails[.....] ← .....
           .....
   ENDIF
ENDFUNCTION
```

[7]

Question 6

7 **(a)** A shirt design company has an order form to order shirts. Customers can order multiple shirts using the same form.

The customer details section has the data:

- name
- address
- telephone number.

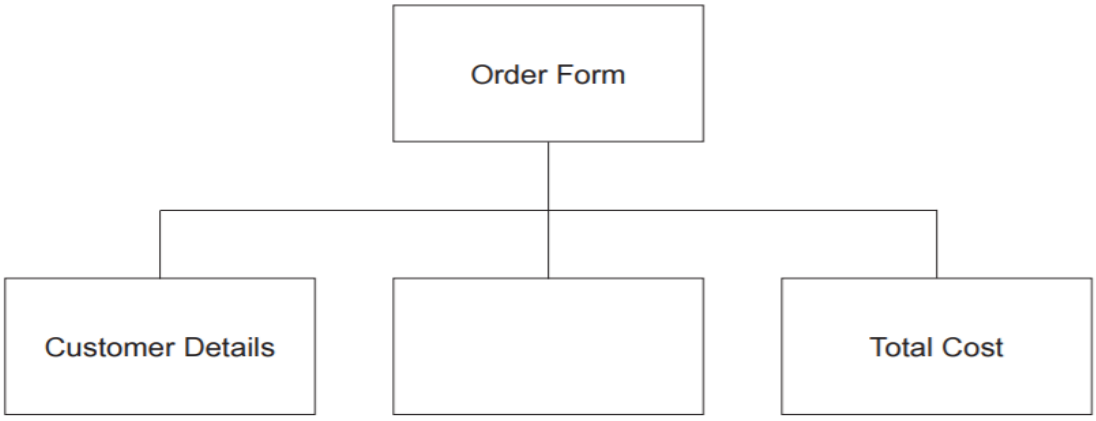
The order details section has the data:

- shirt ID
- colour
- cost.

A total cost for the order is also calculated.

The cost of each shirt is dependent on the size ordered. The sizes customers can order are small, medium and large.

Complete the following JSP data structure diagram for the order form.



[7]

(b) Each customer's order is stored as a record in a file. The customers' orders are stored in the order in which they arrive in the file and no key field is used.

(i) Identify this type of file structure.

..... [1]

(ii) Identify **two other** types of file structure.

1 .....

2 .....

[2]

(c) The procedure `UpdateTelephone()` allows the shirt company to update the record for a customer's details. The procedure will update the telephone number.

The program stores customer details as a custom data type, `Customer`.

The definition for this data type is:

```
TYPE Customer

    Name : STRING

    Address : STRING

    TelephoneNumber : STRING

ENDTYPE
```

The procedure `UpdateTelephone()` takes the customer record to be updated and the new telephone number as parameters. It then updates the telephone number in the record.

Complete the **pseudocode** for the procedure `UpdateTelephone()`.

```
PROCEDURE UpdateTelephone(..... ThisCustomer : Customer,

          ..... NewTelephoneNumber : STRING)

    .....

ENDPROCEDURE
```

[3]

(d) The shirt company is looking to implement a system to reward customers. The system includes:

- 10% discount on orders over $50
- free gift if order over $50 and if the order is placed on a Monday
- additional 5% discount if a customer has a loyalty card
- free delivery for a customer with a loyalty card and spends over $50.

Complete the following decision table for this system.

Conditions	**Order over $50**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**
	Monday	**Y**	**Y**	**N**	**N**	**Y**	**Y**	**N**	**N**
	Loyalty card	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
Actions	**Additional 5% discount**								
	10% discount								
	Free gift								
	Free delivery								

[4]

Question 7

1 Amar is alerted to a run-time error when he runs a program.

(a) A run-time error occurs when Amar attempts to open a file that does not exist.

State **three other** reasons why a run-time error may occur.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Question 8

2 A business is developing a program that stores each customer's username and password in a hash table.

The hash table will be implemented as a 1D array, `CustomerLogIn`, of the custom data type `CustomerRecord`.

The declaration for `CustomerRecord` is:

```
TYPE CustomerRecord

      DECLARE Username : STRING

      DECLARE Password : STRING

ENDTYPE
```

The hash table will store a maximum of 3000 records. The key field will be `Username`.

(a) The program declares the hash table and initialises the username and password of all the records to an empty string.

Write **pseudocode** to declare **and** initialise the hash table.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) (i) A function, `SearchHashTable()` will search for a customer's record in the hash table.

The function `Hash()`:

- takes a `Username` as a parameter
- performs the hashing algorithm
- returns the calculated index of the username within the hash table.

The function `SearchHashTable()`:

- takes the username to search for as a parameter
- uses `Hash()` to calculate the first index of this username within the hash table
- returns either the index of the username if found, or –1 if not found.

Complete the **pseudocode** for the function `SearchHashTable()`.

```
FUNCTION SearchHashTable(BYVALUE SearchUser : STRING) RETURNS .....
    DECLARE Index : INTEGER
    DECLARE Count : INTEGER
    Index ← ..... (SearchUser)
    Count ← 0
    WHILE (CustomerLogIn[Index] ..... <> .....)
           AND(CustomerLogIn[Index].Username <> "")
           AND(Count < 2999)
        Index ← Index + 1
        Count ← Count + 1
        IF Index > .....
            THEN
                Index ← 0
        ENDIF
    ENDWHILE
        IF CustomerLogIn[Index].Username = .....
            THEN
                RETURN Index
             ELSE
                 RETURN .....
         ENDIF
      ENDFUNCTION
```

[7]

(ii) Explain the purpose of the variable `Count` in the function `SearchHashTable()`.

.....

.....

.....

..... [2]

Question 9

3 Recursive algorithms can be used when creating programs.

(a) Describe what is meant by a **recursive algorithm**.

.....

.....

.....

.....

.....

..... [3]

(b) A string that is a palindrome, reads the same forwards as it does backwards. For example, the name Anna is a palindrome.

The function `Substring(Variable, StartingCharacter, NumberOfCharacters)` returns one or more characters from a string. The first character is at position 0.

For example, the string `"Happy"` is stored in the variable `Word`.

- `Substring(Word, 1, 1)` would return the character `"a"`.
- `Substring(Word, 2, 3)` would return the characters `"ppy"`.

The function `Length()` returns the length of the string as an integer. For example, `Length(Word)` returns `5`.

The following is a recursive function to find out whether a string is a palindrome. The function returns `True` if the parameter is a palindrome, and returns `False` if it is not a palindrome.

Complete the **pseudocode** for the recursive algorithm to indicate whether a string is a palindrome.

```
FUNCTION IsPalindrome(CheckWord : STRING) RETURNS BOOLEAN
    IF ..................................................... <= 1
        THEN
            RETURN ..................................
    ENDIF
    IF Substring(CheckWord, 0, 1) <>
             Substring(CheckWord, ..................................(CheckWord)-1, 1)
        THEN
            RETURN ..................................
        ELSE
            RETURN ..................................(Substring(CheckWord, 1,
                                                    Length(CheckWord)-2))
    ENDIF
ENDFUNCTION
```

[5]

(c) The function `FindPower()` is a recursive function that calculates the result of a base number to the power of the exponent. For example, the result of $2^4 = 16$, as 2*2*2*2 = 16. In this example, 2 is the base number and 4 is the exponent.

The base number and the exponent are passed as parameters.

Write **pseudocode** for the recursive function `FindPower()`. Assume both the base and the exponent are positive integers.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

Question 10

4 **(a)** A tennis club has a booking form to book lessons with an instructor.

Club members can book up to five lessons using the booking form.

The customer details section has the data:

- name
- address
- telephone number.

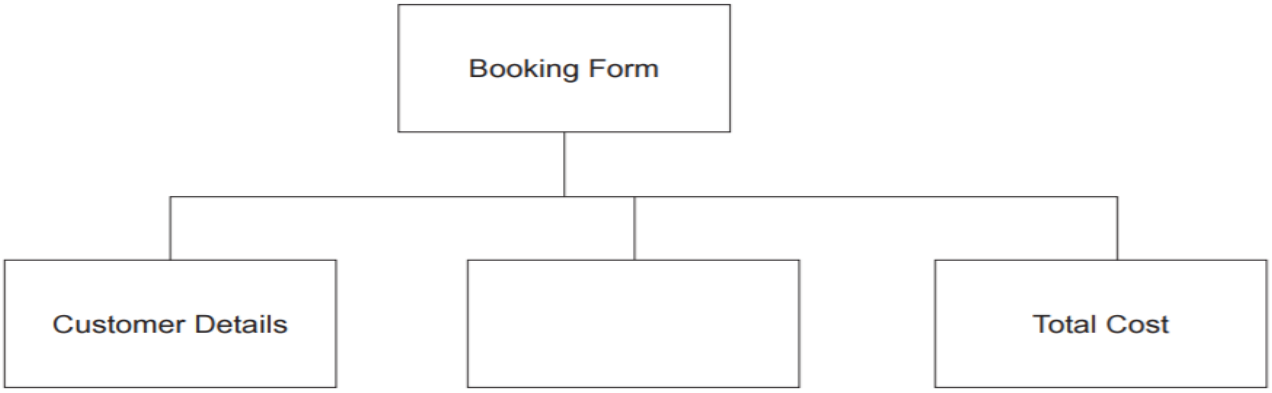
The lesson details section has the data:

- lesson type
- date and time
- lesson cost.

The cost of each lesson is dependent on the customer's type of membership. The membership can be bronze, silver or gold.

The total cost is also calculated.

Complete the following JSP data structure diagram for the booking form.



[7]

(b) State **two** programming constructs that are shown in a JSP data structure diagram.

1 .....

.....

2 .....

.....

[2]

Question 11

5 A declarative programming language is used to represent the following knowledge base.

```
01 person(william).
02 person(deeraj).
03 person(ingrid).
04 person(meghan).
05 country(england).
06 country(spain).
07 country(bangladesh).
08 country(new_zealand).
09 country(malaysia).
10 country(mauritius).
11 visited(william, spain).
12 visited(ingrid, new_zealand).
13 visited(deeraj, spain).
14 visited(meghan, spain).
```

These clauses have the following meanings:

Clause	Meaning
02	Deeraj is a person
05	England is a country
11	William has visited Spain

(a) Gina is a person who has visited Cyprus.

Write additional clauses to represent this information.

15 .....

16 .....

17 .....

[3]

(b) Write the result returned by the goal:

```
visited(X, spain).
```

X = ... [2]

(c) P might visit C, if P is a person, C is a country and P has not visited C.

Write this as a rule.

```
mightvisit(P , C)
```

IF .....

.....

..... [4]

Question 12

6 Object-oriented programming has several features. These include containment, classes, methods and properties.

(a) Describe what is meant by **containment**.

.....

.....

.....

.....

.....

..... [3]

(b) Identify **two other** features of object-oriented programming.

1 .....

2 .....

[2]

Question 13

8 Files can be structured in serial, sequential or random format.

Tick (✓) **one** box in each row to show whether the statement applies to **Serial**, **Sequential** or **Random** format.

Statement	Serial	Sequential	Random
Uses a hashing algorithm			
No key field is used when storing data, for example, it is stored in chronological order			
Collisions can occur			
Least efficient for a very large number of records			
Most efficient for a very large number of records			

[3]

Question 14

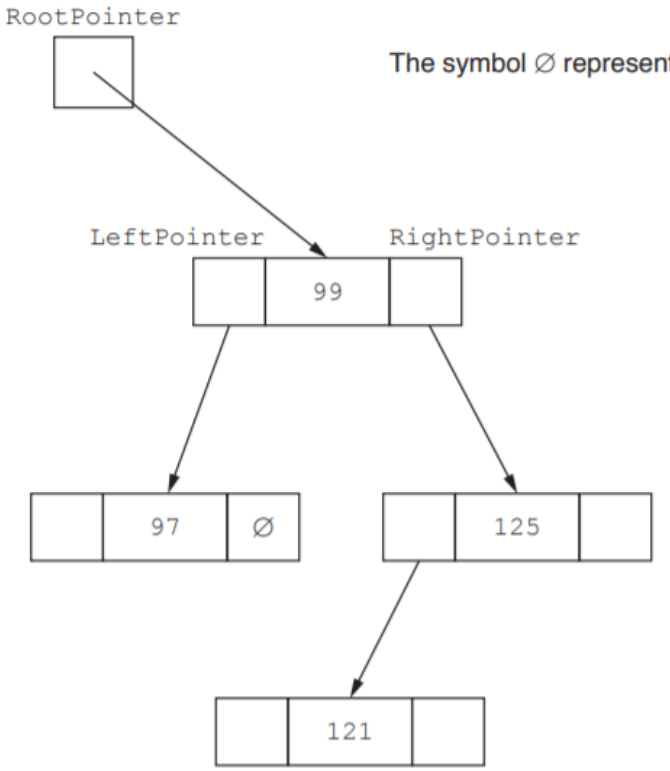
2 A computer games club wants to run a competition. The club needs a system to store the scores achieved in the competition.

A selection of score data is as follows:

99, 125, 121, 97, 109, 95, 135, 149

(a) A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

(i) Complete the binary tree to show how the score data above will be organised.



[5]

(ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from **part (a)(i)**.

RootPointer

0

FreePointer

Index	LeftPointer	Data	RightPointer
0		99	
1		125	
2		121	
3		97	
4		109	
5		95	
6		135	
7		149	
8			

[6]

(b) The club also considers storing the data in the order in which it receives the scores as a linked list in a 1D array of records.

The following pseudocode algorithm searches for an element in the linked list.

Complete the **six** missing sections in the algorithm.

```
FUNCTION FindElement(Item : INTEGER) RETURNS ..........

    .......... ← RootPointer

    WHILE CurrentPointer .......... NullPointer

        IF List[CurrentPointer].Data <> ..........

            THEN

                CurrentPointer ← List[..........].Pointer

            ELSE

                RETURN CurrentPointer

        ENDIF

    ENDWHILE

    CurrentPointer ← NullPointer

    .......... CurrentPointer

ENDFUNCTION
```

[6]

Question 15

(c) The games club is looking at two programming paradigms: imperative and object-oriented programming paradigms.

Describe what is meant by the **imperative programming paradigm** and the **object-oriented programming paradigm**.

(i) Imperative .....

.....

.....

.....

.....

..... [3]

(ii) Object-oriented .....

.....

.....

.....

.....

..... [3]

Question 16

(f) In **part (b)**, the club stored scores in a 1D array. This allows the club to sort the scores.

The following is a sorting algorithm in pseudocode.

```
NumberOfScores ← 5
FOR Item ← 1 TO NumberOfScores - 1
   InsertScore ← ArrayData[Item]
   Index ← Item - 1
   WHILE (ArrayData[Index] > InsertScore) AND (Index >= 0)
      ArrayData[Index + 1] ← ArrayData[Index]
      Index ← Index - 1
   ENDWHILE
   ArrayData[Index + 1] ← InsertScore
ENDFOR
```

(i) Give the name of this algorithm.

..... [1]

(ii) State the name of **one** other sorting algorithm.

..... [1]

(iii) Complete a dry run of the algorithm using the following trace table.

Item	NumberOfScores	InsertScore	Index	ArrayData				
				0	1	2	3	4
				99	125	121	109	115

[7]

Question 17

3 Some algorithms can be written using recursion.

(a) State **two** features of recursion.

Feature 1 .....

Feature 2 .....

[2]

(b) Explain what a compiler has to do to implement recursion.

.....

.....

.....

.....

.....

..... [3]

Question 18

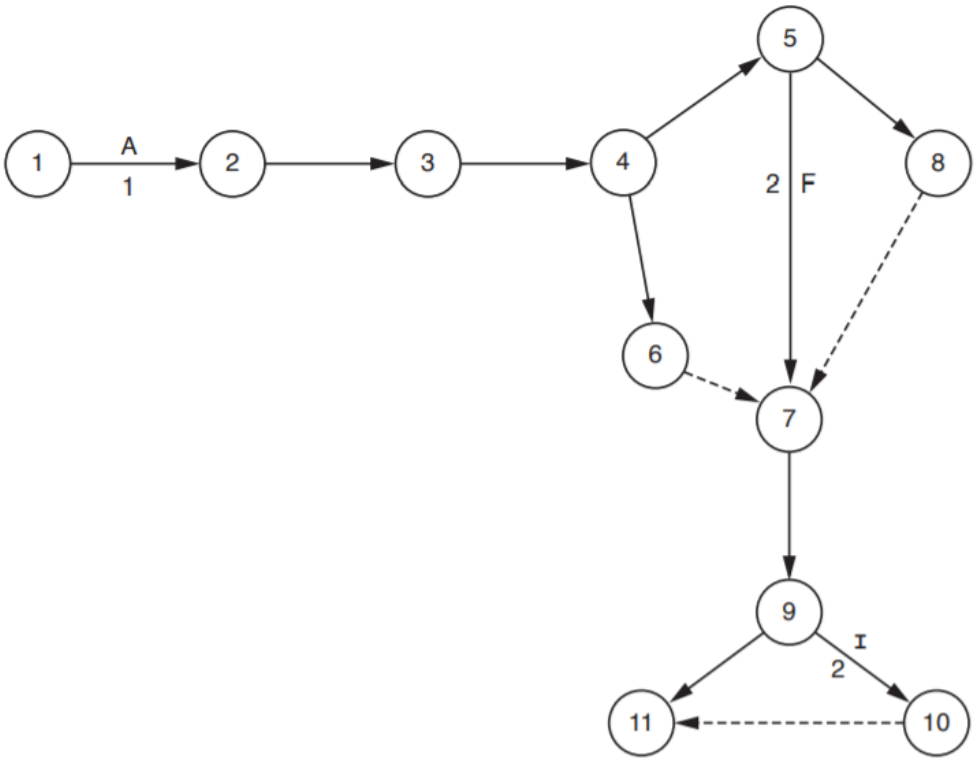
1 A technology company needs software to calculate how much each employee should be paid.

(a) Developing the software will involve the following activities:

Activity	Description	Time to complete (weeks)	Predecessor
A	Identify requirements	1	–
B	Observe current system	1	A
C	Create algorithm design	3	B
D	Write code	10	C
E	Test modules	7	C
F	White box testing	2	D
G	Black box testing	3	D
H	Install software	1	E, F, G
I	Acceptance testing	2	H
J	Create user documentation	2	H

(i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Three of the activities and times have been done for you.



[7]

(ii) The dashed line connecting nodes 10 and 11 indicates a dummy activity.

State the purpose of a dummy activity.

.....

..... [1]

(b) A bonus payment may be added to an employee's salary. A pension payment may also be subtracted from an employee's salary.

The company needs to assess what additions and subtractions should be made to the salary of each employee. There are three conditions to check:

- If the employee has worked a public holiday, they receive a 3% bonus payment.
- If the employee has worked 160 or more hours in a month, they receive an additional 5% bonus payment.
- If the employee pays into a pension, the company subtracts 4% for the pension payment.

Complete the decision table to show the additions and subtractions.

		Rules							
Conditions	**Public holiday**	Y	Y	Y	Y	N	N	N	N
	Hours >= 160	Y	Y	N	N	Y	Y	N	N
	Pension	Y	N	Y	N	Y	N	Y	N
Actions	**3% bonus payment**								
	5% bonus payment								
	4% pension payment								

[3]

Question 19

(d) Noona describes an example of a feature of object-oriented programming (OOP). She says:

"One method exists in the parent class but is overwritten in the child class, to behave differently."

Identify the feature Noona has described.

..... [1]

Question 20

2 The number of cars that cross a bridge is recorded each hour. This number is placed in a circular queue before being processed.

(a) The queue is stored as an array, `NumberQueue`, with eight elements. The function `AddToQueue` adds a number to the queue. `EndPointer` and `StartPointer` are global variables.

Complete the following **pseudocode** algorithm for the function `AddToQueue`.

```
FUNCTION AddToQueue(Number : INTEGER) RETURNS BOOLEAN

   DECLARE TempPointer : INTEGER

   CONSTANT FirstIndex = 0

   CONSTANT LastIndex = .....

   TempPointer ← EndPointer + 1

   IF ..... > LastIndex

       THEN

          TempPointer ← .....

   ENDIF

   IF TempPointer = StartPointer

       THEN

          RETURN .....

       ELSE

          EndPointer ← TempPointer

          NumberQueue[EndPointer] ← .....

          RETURN TRUE

   ENDIF

ENDFUNCTION
```

[5]

(b) Describe how a number is removed from the circular queue to be processed.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) A queue is one example of an Abstract Data Type (ADT).

Identify **three other** Abstract Data Types.

1 .....

2 .....

3 .....

[3]

Question 21

3 A company wants to test a program to check that it works. They can use different types of test data to do this.

(a) Identify **three** different types of test data that the company can use.

1 .....

2 .....

3 .....

[3]

(b) The programmer will make use of debugging features, when building and testing a program.

(i) Two debugging features are described in the table.

Write the correct name for **each** debugging feature.

Description	Name of debugging feature
A point where the program can be halted to see if the program works to this point.	.....
One statement is executed and then the program waits for input from the programmer to move on to the next statement.	.....

[2]

(ii) Identify **and** describe **one other** debugging feature.

Debugging feature .....

Description .....

.....

[2]

Question 22

5 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC) and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

(a) A programmer needs a program that multiplies a binary number by 4.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Write the program using the given instruction set.

Label	Instruction		Comment
	Op code	Operand	
			`// load contents of NUMBER`
			`// perform shift to multiply by 4`
			`// store contents of ACC in NUMBER`
			`// end program`
`NUMBER:`	`B00110110`		

[5]

Note:

`#` denotes immediate addressing
`B` denotes a binary number, e.g. `B01001010`
`&` denotes a hexadecimal number, e.g. `&4A`

(b) A programmer needs a program that counts the number of lower case letters in a string.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Complete the program using the given instruction set. A copy of the instruction set is provided on the opposite page.

Label	Instruction		Comment
	Op code	Operand	
	LDR	#0	// initialise Index Register to 0
START:			// load the next value from the STRING
			// perform bitwise AND operation with MASK
			// check if result is equal to MASK
			// if FALSE, jump to UPPER
			// increment COUNT
UPPER:	INC	IX	// increment the Index Register
			// decrement LENGTH
			// is LENGTH = 0 ?
			// if FALSE, jump to START
	END		// end program
MASK:	B00100000		// if bit 5 is 1, letter is lower case
COUNT:	0		
LENGTH:	5		
STRING:	B01001000		// ASCII code for 'H'
	B01100001		// ASCII code for 'a'
	B01110000		// ASCII code for 'p'
	B01110000		// ASCII code for 'p'
	B01011001		// ASCII code for 'Y'

[8]

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

Question 23

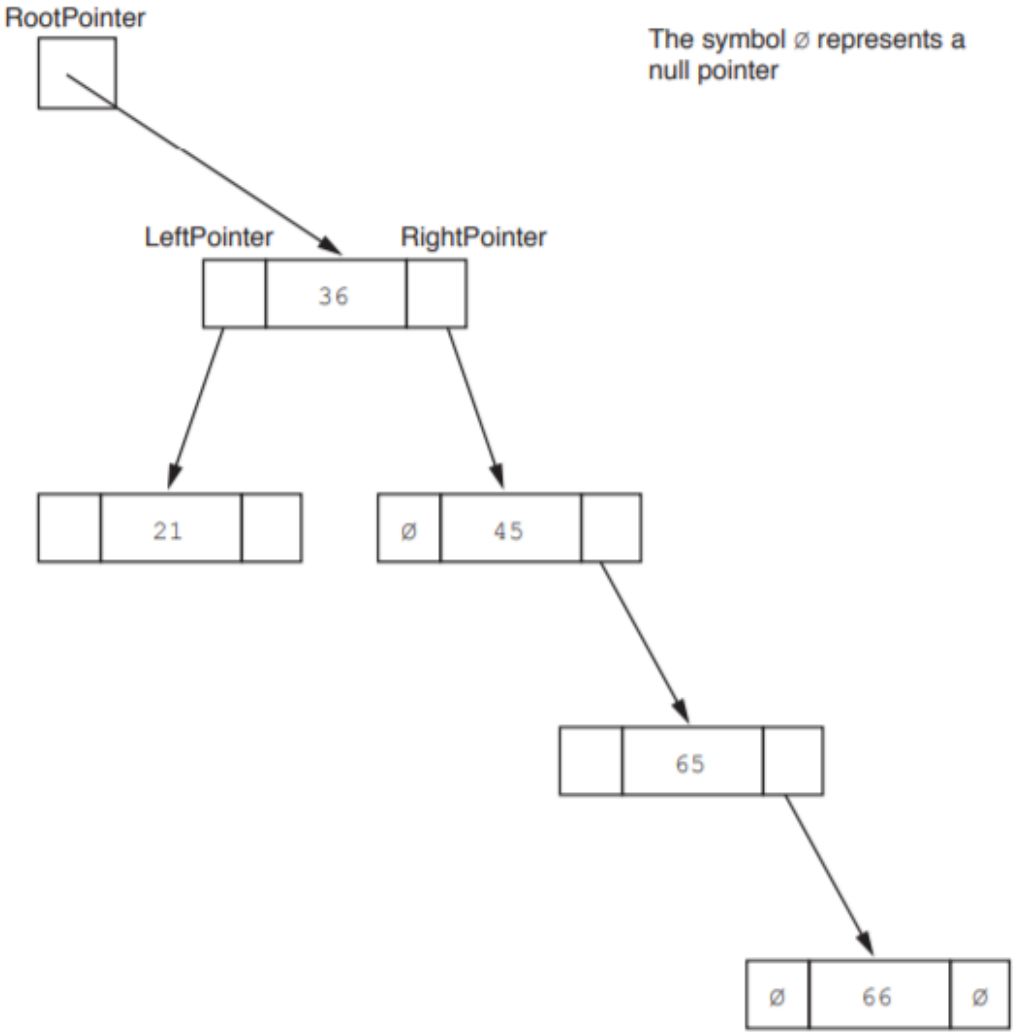
1 A company wants an online marking system for an examination.

(a) The following is a selection of data showing final marks.

```
36, 45, 21, 65, 66, 13, 54, 53, 34
```

A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

(i) Complete the binary tree to show how the data above will be organised.



[5]

(ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from **part (a)(i)**.

RootPointer

0

FreePointer

Index	LeftPointer	Data	RightPointer
0		36	
1		45	
2		21	
3		65	
4		66	
5		13	
6		54	
7		53	
8		34	
9			

[6]

Question 24

(ii) Get and set methods are used to support the security and integrity of data in object-oriented programming.

Explain how get and set methods are used to support security and integrity.

.....

.....

.....

.....

.....

..... [3]

Question 25

(c) The examination paper will be taken by many candidates in centres around the world.

The program stores the objects of the `ExaminationPaper` class in a file. The company has decided to use a hash table, rather than a linked list to store the objects.

Explain why a hash table is more suitable than a linked list to store the objects.

.....

.....

.....

.....

.....

.....

.....

..... [4]

Question 26

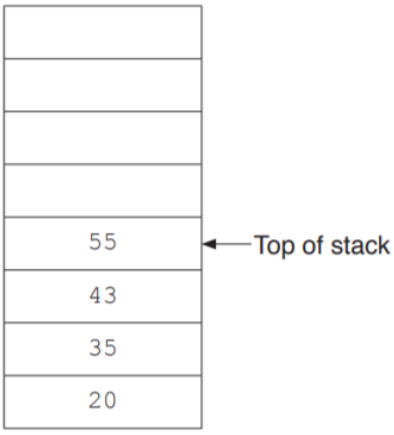
2 A stack is an Abstract Data Type (ADT).

(a) Tick (✓) **one** box to show the statement that describes a stack data structure.

Statement	Tick (✓)
Last in first out	
First in first out	
Last in last out	

[1]

(b) A stack contains the values `20`, `35`, `43`, `55`.



(i) Show the contents of the stack in **part (b)** after the following operations.

```
POP()

POP()

PUSH(10)
```

[1]

(ii) Show the contents of the stack from **part (b)(i)** after these further operations:

```
POP()

PUSH(50)

PUSH(55)

POP()

PUSH(65)
```

[1]

(iii) The stack is implemented as a 1D array, with eight elements, and given the identifier `ArrayStack`.

The global variable `Top` contains the index of the last element in the stack, or –1 if the stack is empty.

The function `Push()`:

- takes as a parameter an `INTEGER` value to place on the stack
- adds the value to the top of the stack and returns `TRUE` to show that the operation was successful
- returns `FALSE` if the stack is full.

Write an algorithm in **pseudocode** for the function `Push()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

Question 27

3 **(a)** Identify **and** describe **two** features of an editor that can help a programmer to write program code.

Feature 1 .....

Description .....

.....

.....

Feature 2 .....

Description .....

.....

.....

[4]

(b) A programmer can use three types of test data when testing a program.

Identify the **three** different types of test data.

1 .....

2 .....

3 .....

[3]

Question 28

4 **(a)** A program has sorted some data in the array, `List`, in ascending order.

The following binary search algorithm is used to search for a value in the array.

```
01   ValueFound ← FALSE
02   UpperBound ← LengthOfList - 1
03   LowerBound ← 0
04   NotInList ← FALSE
05
06   WHILE ValueFound = FALSE AND NotInList = FALSE
07      MidPoint ← ROUND((LowerBound + UpperBound) / 2)
08
09      IF List[LowerBound] = SearchValue
10         THEN
11            ValueFound ← TRUE
12         ELSE
13            IF List[MidPoint] < SearchValue
14               THEN
15                  UpperBound ← MidPoint + 1
16               ELSE
17                  UpperBound ← MidPoint - 1
18            ENDIF
19            IF LowerBound > MidPoint
20               THEN
21                  NotInList ← TRUE
22            ENDIF
23      ENDIF
24   ENDWHILE
25
26   IF ValueFound = FALSE
27      THEN
28         OUTPUT "The value is in the list"
29      ELSE
30         OUTPUT "The value is not found in the list"
31   ENDIF
```

Note:

The pseudocode function

```
ROUND(Reall : REAL) RETURNS INTEGER
```

rounds a number to the nearest integer value.

For example: `ROUND(4.5)` returns `5` and `ROUND(4.4)` returns `4`

(i) There are four errors in the algorithm.

Write the line of code where an error is present **and** write the correction in **pseudocode**.

Error 1 .....

Correction .....

Error 2 .....

Correction .....

Error 3 .....

Correction .....

Error 4 .....

Correction .....

[4]

(ii) A binary search is one algorithm that can be used to search an array.

Identify another searching algorithm.

..... [1]

(b) The following is an example of a sorting algorithm. It sorts the data in the array `ArrayData`.

```
01  TempValue ← ""
02  REPEAT
03      Sorted ← TRUE
04      FOR Count ← 0 TO 4
05          IF ArrayData[Count] > ArrayData[Count + 1]
06              THEN
07                  TempValue ← ArrayData[Count + 1]
08                  ArrayData[Count + 1] ← ArrayData[Count]
09                  ArrayData[Count] ← TempValue
10                  Sorted ← FALSE
11          ENDIF
12      ENDFOR
13  UNTIL Sorted = TRUE
```

(i) Complete the trace table for the algorithm given in **part (b)**, for the `ArrayData` values given in the table.

Count	TempValue	Sorted	ArrayData					
			0	1	2	3	4	5
			5	20	12	25	32	29

[4]

(ii) Rewrite lines 4 to 12 of the algorithm in **part (b)** using a `WHILE` loop instead of a `FOR` loop.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(iii) Identify the algorithm shown in **part (b)**.

..... [1]

(iv) Identify another sorting algorithm.

..... [1]

Question 29

1 Each student at CIE University needs a printing account to print documents from university computers.

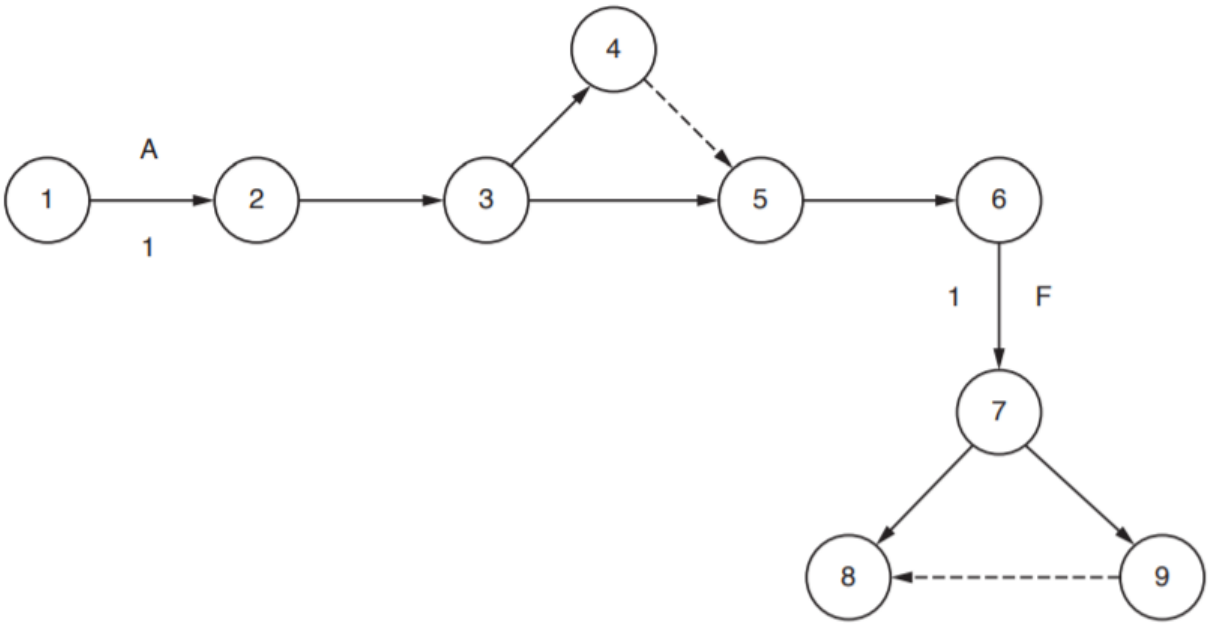
The university is developing software to manage each student's printing account and the printing process.

(a) Developing the software will include the following activities.

Activity	Description	Time in weeks	Predecessor
A	Identify requirements	1	-
B	Produce design	3	A
C	Write code	10	B
D	Test modules	7	B
E	Final system black-box testing	3	C, D
F	Install software	1	E
G	Acceptance testing	2	F
H	Create user documentation	2	F

(i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Two activities and times have been done for you.



[6]

(ii) State what is meant by the **critical path** in a PERT chart.

.....

..... [1]

(iii) Identify **and** describe a project planning technique, other than a PERT chart.

.....

.....

.....

..... [2]

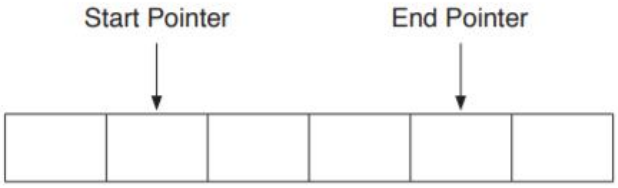
(b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

(i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

(ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.

[3]

(iii) State what would happen if another print job is added to the queue in the status in **part (b)(ii)**.

.....

..... [1]

(iv) The queue is stored as an array, `Queue`, with six elements. The following algorithm removes a print job from the queue and returns it.

Complete the following **pseudocode** for the function `Remove`.

```
FUNCTION Remove RETURNS STRING
   DECLARE PrintJob : STRING
   IF ..... = EndPointer
      THEN
         RETURN "Empty"
      ELSE
         PrintJob ← Queue[.....]
         IF StartPointer = .....
            THEN
               StartPointer ← .....
            ELSE
               StartPointer ← StartPointer + 1
         ENDIF
         RETURN PrintJob
   ENDIF
ENDFUNCTION
```

[4]

(v) Explain why the circular queue could not be implemented as a stack.

.....

.....

.....

..... [2]

Question 30

(d) The university wants to assess troubleshooting issues with a printer. It wants to use a decision table to do this.

The troubleshooting actions are:

- check the connection from computer to printer, if the error light is flashing **and** the document has not been printed
- check the ink status, if the quality is poor
- check whether there is a paper jam, if the error light is flashing **and** the document has not been printed
- check the paper size selected, if the paper size is incorrect.

(i) Describe the purpose of a decision table.

.....

.....

.....

..... [2]

(ii) Complete the rules for the actions in the following decision table.

		Rules							
Conditions	Document printed but the quality is poor	Y	Y	Y	Y	N	N	N	N
	Error light is flashing on printer	Y	Y	N	N	Y	Y	N	N
	Document printed but paper size is incorrect	Y	N	Y	N	Y	N	Y	N
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[4]

(iii) Simplify your solution by removing redundancies.

		Rules							
Conditions	Document printed but the quality is poor								
	Error light is flashing on printer								
	Document printed but paper size is incorrect								
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[5]

Question 31

2 The following table shows part of the instruction set for a processor, which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer writes a program that multiplies two numbers together and outputs the result. The numbers are stored as NUMONE and NUMTWO.

The programmer has started to write the program in the following table. The comment column contains explanations for some of the missing program instructions and data.

Complete the program using the given instruction set.

Label	Op code	Operand	Comment
LOOP:			// load the value from ANSWER
			// add the value from NUMONE
			// load the value from COUNT
			// increment the Accumulator
			// is NUMTWO = COUNT ?
			// if false, jump to LOOP
			// load the value from ANSWER
			// output ANSWER to the screen
			// end of program
NUMONE:	2		
NUMTWO:	4		
COUNT:	0		
ANSWER:	0		

[9]

Question 32

3 Software may not perform as expected. One reason for this is that a syntax error exists in the code.

Identify **three other** reasons why software may not perform as expected.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Question 33

4 The following table contains definitions related to testing terminology.

Complete the table with the correct testing term to match the definition.

Definition	Term
Software is tested by an in-house team of dedicated testers.	.....
Software is tested by the customer before it is signed off.	.....
Software is tested by a small selection of users before general release.	.....

[3]

Question 35

1 A declarative language is used to represent facts and rules about flights.

```
01 direct(edinburgh, paris).
02 direct(palma, rome).
03 direct(glasgow, palma).
04 direct(glasgow, vienna).
05 direct(glasgow, salzburg).
06
07 flies(paris, fly_jet).
08 flies(mumbai, british_air).
09 flies(palma, ciebe).
10 flies(vienna, fly_jet).
11 flies(salzburg, ciebe).
12
13 can_fly(X, Y) IF direct(X, Z) AND direct(Z, Y).
```

These clauses have the following meaning:

Clause	Explanation
01	There is a direct route from Edinburgh to Paris.
07	Fly Jet operates flights to Paris.
13	It is possible to fly from X to Y if there is a direct flight from X to Z and a direct flight from Z to Y.

(a) More facts need to be included.

There is a direct flight from London to Rome and British Air flies to Rome.

14 .....

15 .....

[2]

(b) Using the variable `Q`, the goal

```
flies(Q, fly_jet).
```

returns

```
Q = paris, vienna
```

Write the result returned by the goal

```
flies(K, ciebe).
```

K = ...[2]

(c) Use the variable `M` to write the goal to find where you can fly direct from Glasgow.

.....[2]

(d) If an airline flies to an airport, that airline also flies every direct route out of that airport.

Write a rule to represent this condition.

```
flies(Y, X)
```

IF .....

.....[3]

(e) State what the following goal returns.

```
can_fly(glasgow, rome).
```

.....[1]

Question 36

2 The array `ItemList[1:20]` stores data. A **bubble sort** sorts these data.

(a) Complete the pseudocode algorithm for a bubble sort.

```
01   MaxIndex ← 20
02   NumberItems ← .....
03   FOR Outer ← 1 TO .....
04      FOR Inner ← 1 to NumberItems
05         IF ItemList[Inner] > .....
06            THEN
07               Temp ← ItemList[.....]
08               ItemList[Inner] ← ItemList[.....]
09               ItemList[Inner + 1] ← .....
10         ENDIF
11      ENDFOR
12      NumberItems ← .....
13   ENDFOR
```

[7]

(b) The algorithm in **part (a)** is inefficient.

(i) Explain why the algorithm in **part (a)** is inefficient.

.....

.....

.....

.....[2]

(ii) Explain how you would improve the efficiency of this algorithm.

.....

.....

.....

.....

.....[3]

(c) An insertion sort is another sorting algorithm.

State **two** situations when an insertion sort is more efficient than a bubble sort. Give a reason for each.

Situation 1 .....

.....

Reason .....

.....

.....

Situation 2 .....

.....

Reason .....

.....

.....

[4]

Question 37

3 An internet based music streaming service provides access to an unlimited number of songs for members to play.

The following pseudocode represents the operation of the service.

```
CALL OpenAccount()
CALL OperateAccount()
CALL CloseAccount()

PROCEDURE OperateAccount()
   WHILE RequestCloseAccount() = FALSE
      IF SubscriptionDue() = TRUE
         THEN
            CALL MakePayment()
         ELSE
             CALL PlaySong()
      ENDIF
   ENDWHILE
ENDPROCEDURE
```

(a) Complete the JSP structure diagram for this music service from the pseudocode given.

[5]

(b) The service needs extending so that members can download songs to play offline.

- When a member selects a song, the service checks if the song has already been downloaded.
- If the member has already downloaded the song, the member has the option to delete or play it.
- If the member has not already downloaded the song they have the option to download or stream it.

Complete the following JSP structure diagram to represent these new requirements.

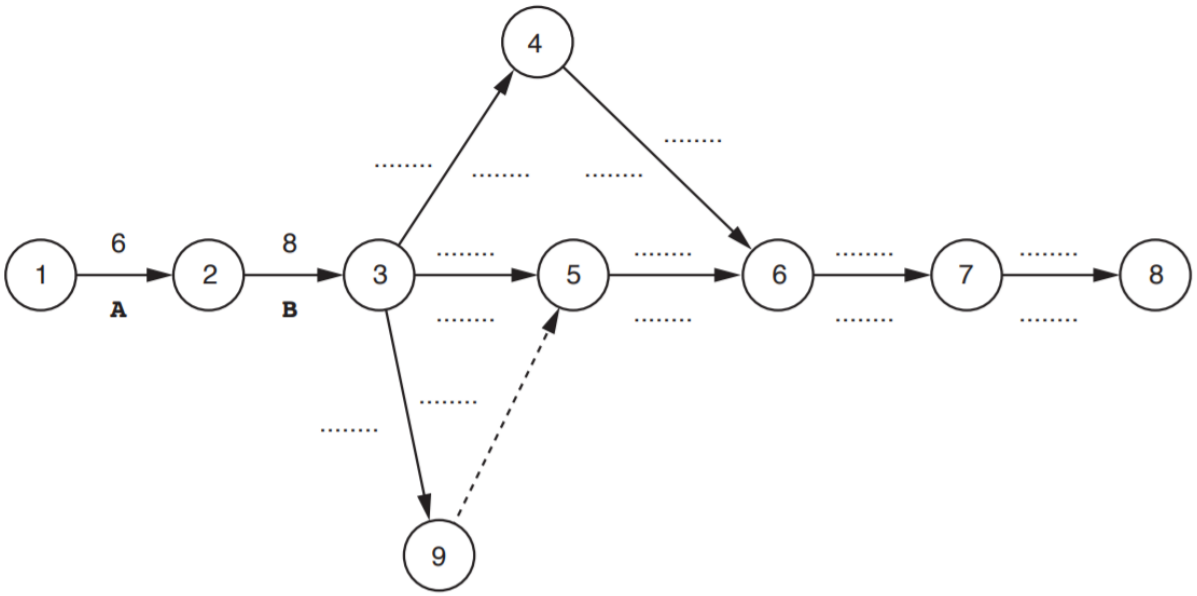
[4]

Question 38

4 A software company is developing a new application. The project manager has created a work breakdown structure, as shown in the following table.

	Activity	Days to complete	Predecessor
A	Gather user requirements	6	
B	Design work	8	A
C	Develop server code	4	B
D	Develop application code	5	B
E	User interface development	6	B
F	Test server code	2	C
G	Test application	2	D,E
H	Test application/server integration	5	F,G
I	Roll out mobile application	3	H

(a) Use the data in the table to complete the following Program Evaluation Review Technique (PERT) chart.



[5]

(b) Calculate the critical path (CP). State the:

activities that form the CP .....

duration of the CP .....

[2]

(c) For activity F, state the:

earliest start time .....

latest finish time .....

[2]

Question 39

6 An Abstract Data Type (ADT) is used to create a linked list. The linked list is created as an array of records. The records are of type `ListNode`.

An example of a record of `ListNode` is shown in the following table.

Data Field	Value
Player	"Alvaro"
Pointer	1

(a) (i) Use **pseudocode** to write a definition for the record type, `ListNode`.

.....

.....

.....

.....

.....[3]

(ii) An array, `Scorers`, will hold 10 nodes of type `ListNode`. Use **pseudocode** to write an array declaration for this array. The lower bound subscript is `0`.

.....[2]

(b) The linked list stores `ListNode` records in alphabetical order of player. The last node in the linked list always has a `Pointer` value of `-1`. The position of the first node in the linked list is held in the variable `ListHead`.

After some processing, the array and variables are in the state as follows:

ListHead
0

Scorers

	Player	Pointer
0	"Alvaro"	1
1	"Antoine"	3
2	"Dimitri"	7
3	"Cristiano"	2
4	"Gareth"	5
5	"Graziano"	6
6	"Olivier"	8
7	"Erik"	4
8	"Yaya"	9
9	"Zoto"	-1

A **recursive** function traverses the linked list to search for a player.

An example of calling the function, using pseudocode, is:

```
Position ← SearchList("Gareth", ListHead)
```

Complete the following **pseudocode** to implement the function `SearchList()`.

The function will return a value of `99` when a player is not found.

```
FUNCTION SearchList(Find : STRING, Position : INTEGER) RETURNS INTEGER
   IF Scorer[Position].Player = ..........
      THEN
         RETURN ..........
      ELSE
         IF Scorer[Position].Pointer <> -1
            THEN
               Position ← SearchList(Find, ..........)
               RETURN ..........
            ELSE
               RETURN ..........
         ENDIF
   ENDIF
ENDFUNCTION
```

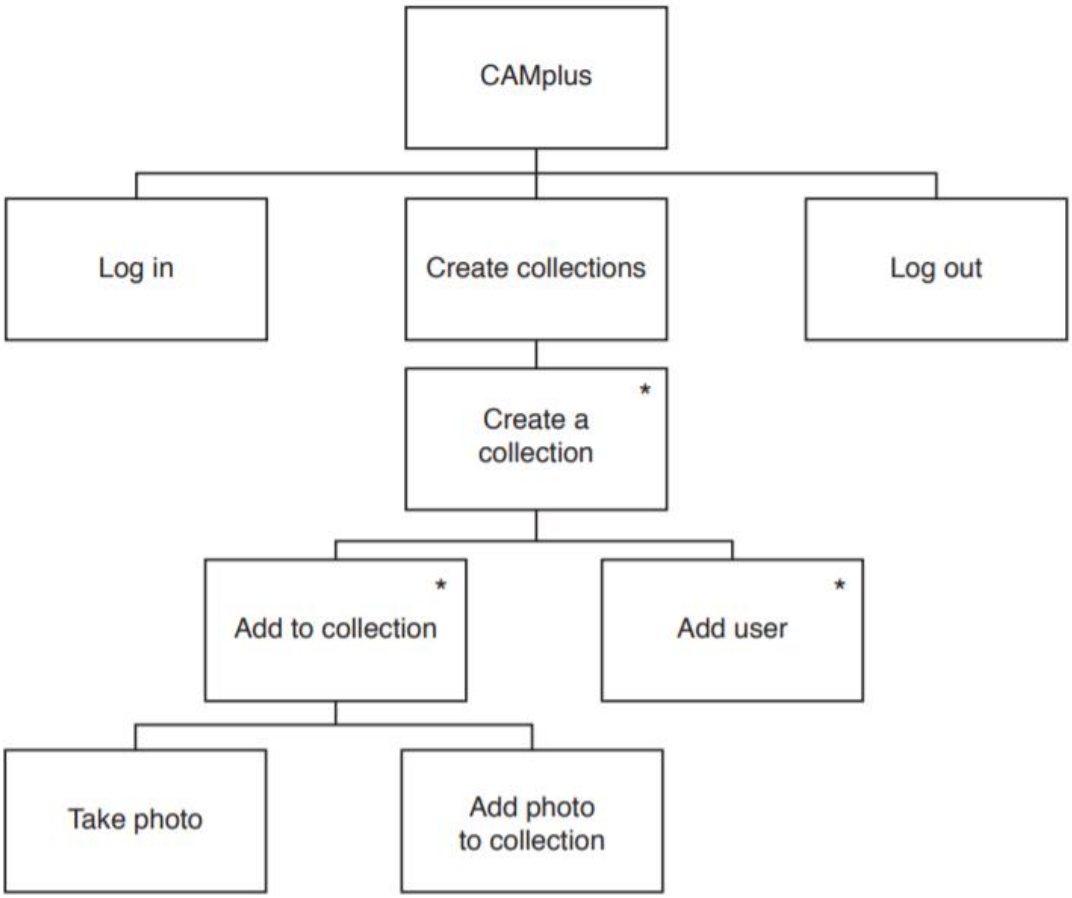
[5]

Question 40

1 Paul is using an application (app) called CAMplus. The app allows users to:

- log in
- create a new collection of photographs
- use the camera to take new photographs
- automatically add new photographs to the new collection
- share the new collection with other users
- start another collection or log out of the app.

The following JSP structure diagram represents the operation of CAMplus.



(a) An algorithm has been written in pseudocode to represent the **Create collections** operation from the JSP structure diagram. The algorithm is incomplete.

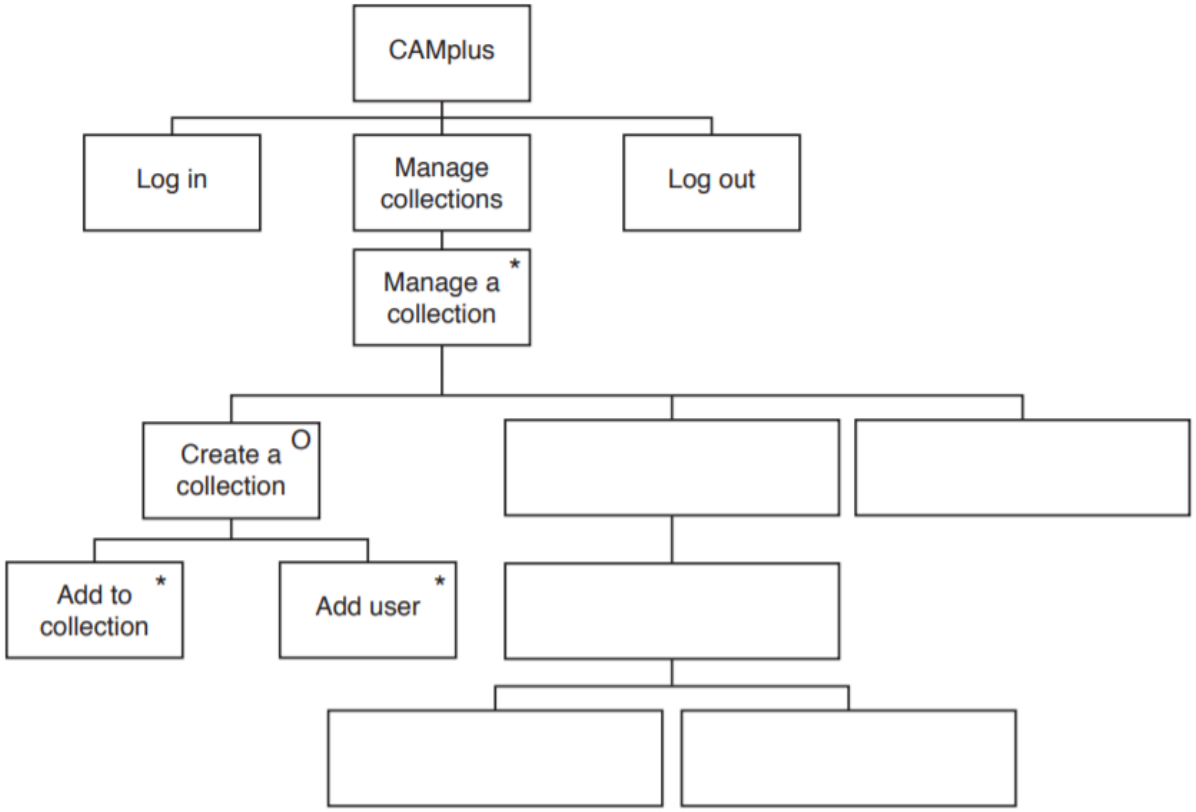
Write **pseudocode** to complete this algorithm.

```
REPEAT

   REPEAT

      CALL TakePhoto

      ..........

      OUTPUT "Do you want to take another photo?"

      INPUT AddPhoto

   UNTIL AddPhoto = "No"

   REPEAT

      ..........

      OUTPUT "Do you want to add another user?"

      INPUT NewUser

   UNTIL ..... = "No"

   OUTPUT "Do you want to create another collection?"

   ..........

UNTIL NewCollection = "No"
```

[4]

(b) The app is updated. Paul can now add and delete photos from chosen collections. Paul can also delete collections.

Complete the JSP structure diagram to show the changes.



Question 41

2 A declarative language is used to represent the following facts and rules about iguanas and lizards.

```
01   has(reptile, cold_blood).
02   has(reptile, air_breathing).
03   has(reptile, scales).
04
05   is_a(squamata, reptile).
06   is_a(iguana, squamata).
07   is_a(lizard, squamata).
08   is_a(green_iguana, iguana).
09   is_a(cayman, iguana).
10   is_a(smooth_iguana, iguana).
11
12   maxsize(green_iguana, 152).
13   maxsize(cayman, 90).
14   maxsize(smooth_iguana, 70).
```

These clauses have the following meaning:

Clause	Explanation
01	A reptile has cold blood.
09	A cayman is a type of iguana.
12	The maximum size of a green iguana is 152 cm.

(a) More facts are to be included.

A gecko is a type of lizard. It has a maximum size of 182 cm.

Write the additional clauses to record these facts.

15 .....

16 .....

[2]

(b) Using the variable `R`, the goal

```
is_a(R, squamata).
```

returns

```
R = iguana, lizard
```

Write the result returned by the goal

```
is_a(T, iguana).
```

`T =` .....[2]

(c) Write the goal, using the variable `X`, to find what a squamata is.

.....[2]

(d) All iguanas and lizards are squamata. All squamata are reptiles.

Write a recursive rule to make all lizards and iguanas inherit the properties of reptiles.

```
has(X, Y)

IF
```

.....

.....[3]

(e) State what the following goal returns.

```
NOT(maxsize(cayman, 70)).
```

.....[1]

Question 42

3 The arrays `PollData[1:10]` and `CardData[1:10]` store data.

PollData	12	85	52	57	25	11	33	59	56	91

CardData	11	12	25	33	52	56	57	59	91	85

An **insertion sort** sorts these data.

(a) State why it will take less time to complete an insertion sort on `CardData` than on `PollData`.

.....

.....[1]

(b) The following pseudocode algorithm performs an insertion sort on the `CardData` array.

Complete the following **pseudocode** algorithm.

```
01  ArraySize ← 10
02  FOR Pointer ← 2 TO .....
03    ValueToInsert ← CardData[Pointer]
04    HolePosition ← .....
05    WHILE (HolePosition > 1 AND (..... > ..... ))
06      CardData[HolePosition] ← CardData[.....]
07      HolePosition ← .....
08    ENDWHILE
09    CardData[HolePosition] ← .....
10  ENDFOR
```

[7]

(c) **(i)** A binary search algorithm is used to find a specific value in an array.

Explain why an array needs to be sorted before a binary search algorithm can be used.

.....

.....

.....

.....

.....

.....

.....

.....[2]

(ii) The current contents of `CardData` are shown.

11	12	25	33	52	56	57	59	85	91

Explain how a binary search will find the value `25` in `CardData`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

(d) Complete this procedure to carry out a binary search on the array shown in **part (c)(ii)**.

```
PROCEDURE BinarySearch(CardData, SearchValue)
   DECLARE Midpoint : INTEGER
   First ← 1
   Last ← ARRAYLENGTH(.....)
   Found ← FALSE
   WHILE (First <= Last) AND NOT(Found)
      Midpoint ← .....
      IF CardData[Midpoint] = SearchValue
         THEN
            Found ← TRUE
         ELSE
            IF SearchValue < CardData[Midpoint]
               THEN
                  Last ← .....
               ELSE
                  First ← .....
            ENDIF
         ENDIF
   ENDWHILE
ENDPROCEDURE
```

[4]

Question 43

5 A company is developing an application program. The project manager has been asked to create a work breakdown schedule for the project as follows:

	Activity	Days to complete	Predecessor activity
A	Gather User Requirements	6	
B	Design work	4	A
C	Develop server code	4	B
D	Develop application code	5	B
E	User Interface Development	6	B
F	Test server code	2	C
G	Test application	2	D, E
H	Test application/server integration	6	F, G
I	Roll out mobile application	6	H

(a) A GANTT chart is created from the work breakdown schedule. Activities **A** and **B** have already been added to the chart.

Complete the GANTT chart.

Activity																													
A																													
B																													
C																													
D																													
E																													
F																													
G																													
H																													
I																													
Day number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**

[5]

(b) State which activities can run in parallel on the following days.

(i) Day 14

.....[1]

(ii) Day 16

.....[1]

(c) Explain how the project manager will use the GANTT chart to make sure the project is completed on time.

.....

.....

.....

.....[2]

Question 44

6 An Abstract Data Type (ADT) is used to create an unordered binary tree. The binary tree is created as an array of nodes. Each node consists of a data value and two pointers.

A record type, `Node`, is declared using pseudocode.

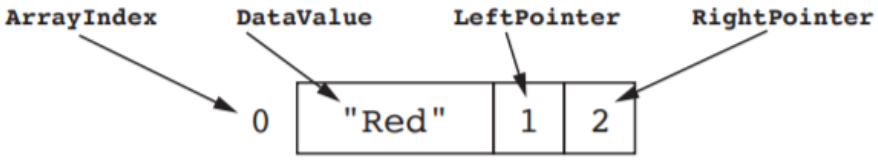
```
TYPE Node
   DECLARE DataValue : STRING
   DECLARE LeftPointer : INTEGER
   DECLARE RightPointer : INTEGER
ENDTYPE
```

The following statement declares an array `BinaryTree`.

```
   DECLARE BinaryTree : ARRAY[0:14] OF Node
```

A variable, `NextNode`, points to the next free node.

The following diagram shows a possible node.



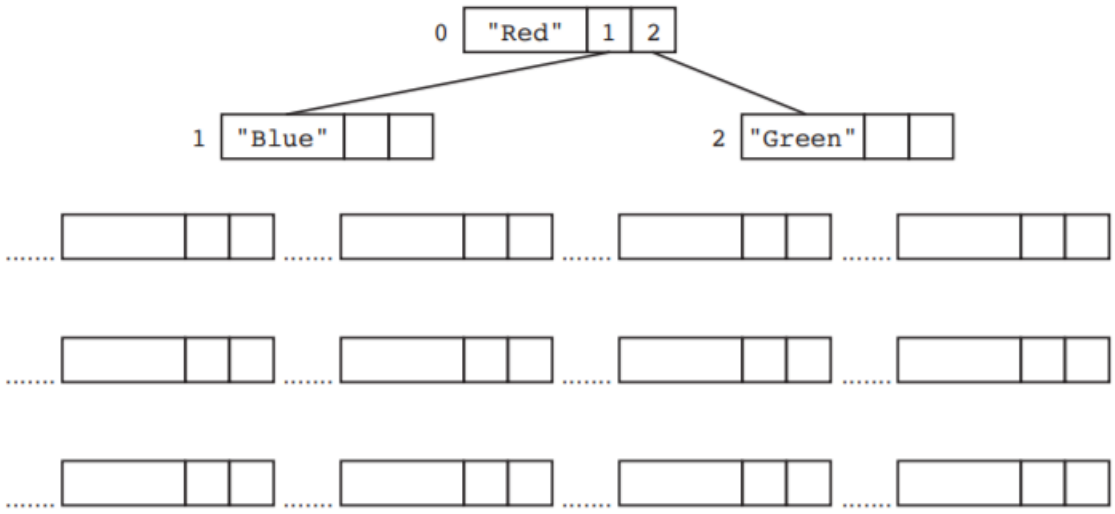
The commands in the following table create and add nodes to the binary tree.

Command	Comment
`CreateTree(NodeData)`	Sets `NextNode` to `0`. Writes `NodeData` into `DataValue` at the position `NextNode` Updates `NextNode` using `NextNode = NextNode + 1`
`AttachLeft(NodeData, ParentNode)`	Writes `NodeData` into `DataValue` of `NextNode` Sets the `LeftPointer` of node `ParentNode` to `NextNode` Updates `NextNode` using `NextNode = NextNode + 1`
`AttachRight(NodeData, ParentNode)`	Writes `NodeData` into `DataValue` of `NextNode` Sets the `RightPointer` of node `ParentNode` to `NextNode` Updates `NextNode` using `NextNode = NextNode + 1`

(a) The following commands are executed.

```
CreateTree("Red")
AttachLeft("Blue", 0)
AttachRight("Green", 0)
```

The following diagram shows the current state of the binary tree.



Write on the diagram to show the state of the binary tree after the following commands have been executed.

```
AttachRight("Black", 2)
AttachLeft("Brown", 2)
AttachLeft("Peach", 3)
AttachLeft("Yellow", 1)
AttachRight("Purple", 1)
AttachLeft("White", 6)
AttachLeft("Pink", 7)
AttachLeft("Grey", 9)
AttachRight("Orange", 9)
```

[5]

Question 45

1 A declarative language is used to represent the following facts and rules about animals.

```
01 feature(dog, drinks_milk).

02 feature(dog, has_lungs).

03 feature(horse, has_lungs).

04 feature(tuna, lives_in_water).

05 feature(tuna, has_gills).

06 feature(crab, lives_in_water).

07 mammal(drinks_milk).

08 mammal(has_lungs).

09 fish(lives_in_water).

10 fish(has_gills).

11 is_a_mammal(X) IF (feature(X, Y) AND mammal(Y)) AND (feature(X, Z)
   AND mammal(Z)).
```

These clauses are explained in the following table.

Clause	Explanation
01	A dog has the feature, drinks milk
07	A mammal drinks milk
11	X is a mammal, if: • X has the feature Y and a mammal has a feature Y, **and** • X has the feature Z and a mammal has the feature Z

(a) More facts are to be included.

(i) A bird has wings, and a bird lays eggs.

Write the additional clauses to record these facts.

12 .....

13 .....

[2]

(ii) An eagle has all the features of a bird.

Write the additional clauses to record this fact.

14 .....

15 .....

[2]

(b) **(i)** Using the variable `B`, the goal

```
feature(B, drinks_milk)
```

returns

```
B = dog
```

Write the result returned by the goal

```
feature(B, lives_in_water)
```

B = ... [2]

(ii) Write a goal, using the variable `C`, to find the feature(s) of tuna.

..... [2]

(c) An animal is a bird if it lays eggs **and** it has wings.

Complete the following rule.

`is_a_bird(X) IF` .....

..... [3]

(d) Declarative programming and object-oriented programming are two examples of programming paradigms.

(i) Define the term **programming paradigm**.

.....

..... [1]

(ii) Give **two** examples of programming paradigms, other than declarative and object-oriented programming.

1 .....

2 .....

[2]

Question 46

3 Joseph is taking a toy apart. Each time he removes an item from the toy, he writes the name of the item at the bottom of a paper list. When he rebuilds the toy, he puts the items back together working from the bottom of the list.

Joseph writes a computer program to create the list using a stack, `Parts`.

(a) Describe a stack structure.

.....

..... [1]

(b) The stack is represented as an array in the program, the first element in the array is `[0]`.

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer 5

	StackContents
0	"Screw 1"
1	"Screw 2"
2	"Back case"
3	"Screw 3"
4	"Engine outer"
5	
6	
7	

(i) Describe the purpose of the variable `StackPointer`.

.....

..... [1]

(ii) The procedure `POP()` removes an item from the stack. The procedure `PUSH(<identifier>)` adds an item to the stack.

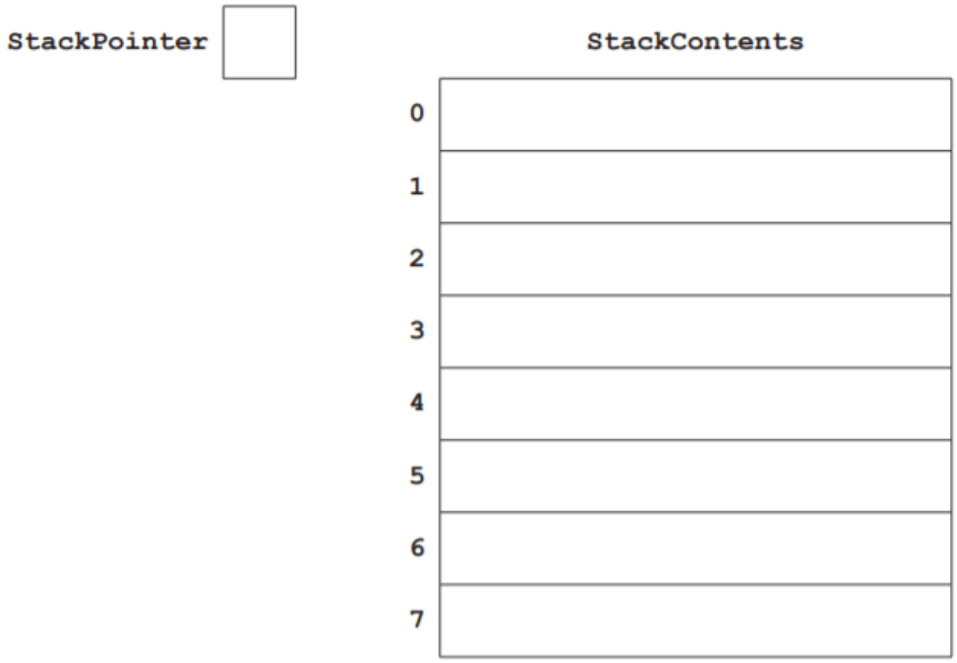
The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer	5

	StackContents
0	"Screw 1"
1	"Screw 2"
2	"Back case"
3	"Screw 3"
4	"Engine outer"
5	
6	
7	

Use the table below to show the contents of the stack, `Parts`, and its pointer after the following code is run.

```
POP()
POP()
PUSH("Light 1")
PUSH("Light 2")
PUSH("Wheel 1")
POP()
POP()
```



[2]

(c) A 1D array, `Parts`, is used to implement the stack. `Parts` is declared as:

```
DECLARE Parts : ARRAY[0 : 19] OF STRING
```

(i) The procedure `POP` outputs the last element that has been pushed onto the stack and replaces it with a `'*'`.

Complete the **pseudocode** for the procedure `POP`.

```
PROCEDURE POP

    IF ..................................... = .....................................

        THEN

            OUTPUT "The stack is empty"

        ELSE

            StackPointer ← ..........................................

            OUTPUT ..................................................................

            Parts[StackPointer] ← ..................................

    ENDIF

ENDPROCEDURE
```

[5]

(ii) The procedure `PUSH()` puts the parameter onto the stack.

Complete the **pseudocode** for the procedure `PUSH()`.

```
PROCEDURE PUSH(BYVALUE Value : String)
    IF StackPointer > .....
       THEN
          OUTPUT "Stack full"
       ELSE
          ..... ← .....
          StackPointer ← .....
    ENDIF
ENDPROCEDURE
```

[4]

Question 47

4 The recursive algorithm for the `Calculate()` function is defined as follows:

```
01 FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER
02     IF Number = 0
03         THEN
04             Calculate ← -10
05         ELSE
06             Calculate ← Number * Calculate(Number - 1)
07     ENDIF
08 ENDFUNCTION
```

(a) (i) State what is meant by a **recursive algorithm**.

.....

..... [1]

(ii) State the line number in `Calculate()` where the recursive call takes place.

..... [1]

(b) The function is called with `Calculate(3)`.

Dry run the function **and** complete the trace table below. State the final value returned. Show your working.

```
01  FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER
02      IF Number = 0
03          THEN
04              Calculate ← -10
05          ELSE
06              Calculate ← Number * Calculate(Number - 1)
07      ENDIF
08  ENDFUNCTION
```

Working .....

.....

.....

.....

Trace table:

Call number	Function call	Number = 0 ?	Return value

Final return value .....

[6]

(c) A recursive algorithm within a subroutine can be replaced with an iterative algorithm.

(i) Describe **one** problem that can occur when running a subroutine that has a recursive algorithm.

.....

.....

.....

..... [2]

(ii) Rewrite the `Calculate()` function in **pseudocode**, using an **iterative algorithm**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

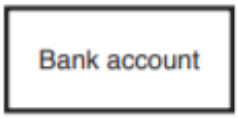
Question 48

1 A bank provides bank accounts to customers.

The following pseudocode represents the operation of the bank account.

```
CALL OpenAccount()
CALL AccountLifeTime()
CALL CloseAccount()

PROCEDURE AccountLifeTime()
   REPEAT
      CALL Transactions()
   UNTIL AccountClosed() = TRUE
ENDPROCEDURE

PROCEDURE CloseAccount()
   IF ReopenAccount() = TRUE
      THEN
         CALL FlagToReopen()
      ELSE
         CALL DeletePermanently()
   ENDIF
ENDPROCEDURE
```

(a) Complete the JSP structure diagram for this bank account from the pseudocode given.



[6]

(b) A transaction can be a credit (deposit) or a debit (withdrawal).

There are two types of transaction that are credits (deposits). These are:

- SWIFT payment
- BACS payment

There are three types of transaction that are debits (withdrawals). These are:

- Debit card payment
- Cheque payment
- Online payment

Complete the JSP structure diagram to represent these additional requirements.

[4]

Question 49

2 A declarative language is used to represent facts and rules about dogs that perform tasks to help people.

```
01 type(pointer, gundog).
02 type(flushing, gundog).
03 type(retriever, gundog).
04
05 is_a(labrador, retriever).
06 is_a(newfoundland, retriever).
07 is_a(cocker_spaniel, flushing).
08 is_a(springer_spaniel, flushing).
09 is_a(king_charles, flushing).
10 is_a(english_setter, pointer).
11 is_a(irish_setter, pointer).
12
13 fav_bird(pointer, grouse).
14 fav_bird(flushing, pheasant).
15 fav_bird(retriever, waterfowl).
16
17 type(X, Y) IF is_a(Z, X) AND type(Z, Y).
```

These clauses have the following meaning:

Clause	Explanation
01	A pointer is a type of gundog.
05	A labrador is an example of a retriever.
13	The favourite bird of a pointer is a grouse.
17	X is a type of Y if Z is an example of X and Z is a type of Y.

(a) More facts are to be included.

A **standard poodle** is an example of a waterdog. A waterdog is a type of gundog.

Write the additional clauses to record these facts.

18 .....

19 .....

[2]

(b) Using the variable `P`, the goal

```
is_a(P, retriever)
```

returns

```
P = labrador, newfoundland
```

Write the result returned by the goal

```
is_a(H, pointer)
```

`H` = ... [2]

(c) Write a query, using the variable `W`, to find out what an **irish setter** is an example of.

..... [2]

(d) `Y` is the favourite bird of dog `X`.

Complete the following rule:

`fav_bird(X, Y) IF` .....

..... [3]

(e) State the value returned by the goal

```
NOT(is_a(labrador, retriever))
```

..... [1]

Question 50

3 A bubble sort algorithm is used to sort an integer array, `List`. This algorithm can process arrays of different lengths.

(a) Write **pseudocode** to complete the bubble sort algorithm shown.

```
01 FOR Outer ← ..... TO 0 STEP - 1
02   FOR Inner ← 0 TO (.....)
03     IF ..... > .....
04       THEN
05         Temp ← .....
06         List[Inner] ← .....
07         List[Inner + 1] ← .....
08     ENDIF
09   ENDFOR
10 ENDFOR
```

[7]

(b) **(i)** State the order of the sorted array.

..... [1]

(ii) State which line of the algorithm you would change to sort the array into the opposite order.

State the change you would make.

Line .....

Change .....

.....

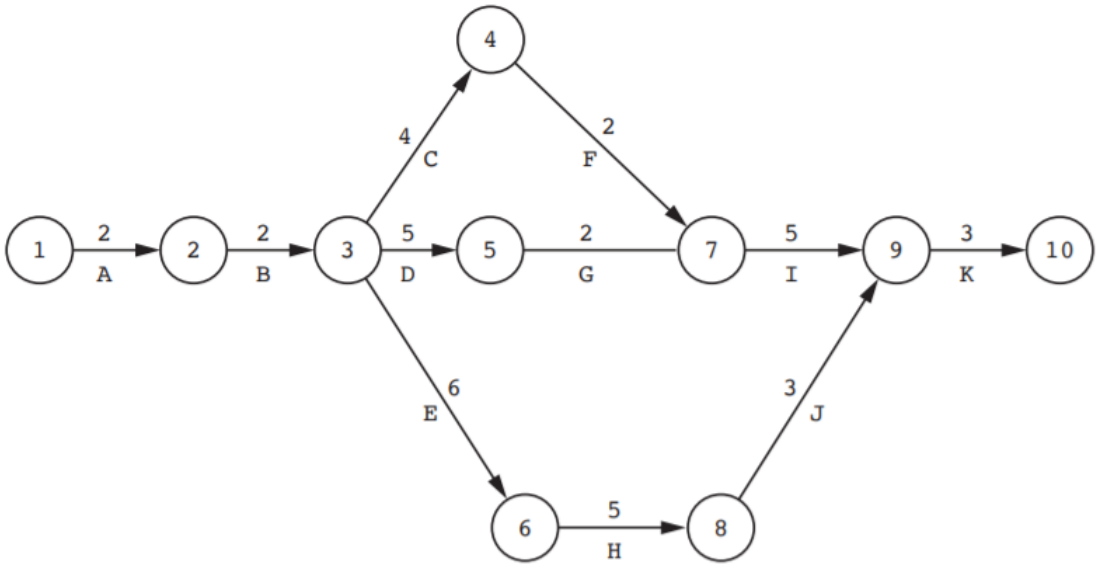
[1]

(c) Use **pseudocode** to write an alternative version of this bubble sort algorithm that will exit the algorithm when the list is fully sorted.

[4]

Question 51

5 A Program Evaluation Review Technique (PERT) chart has been constructed for a project that is at the planning stage.



(a) Complete the following GANTT chart using the information in the PERT chart.

A																								
B																								
C																								
D																								
E																								
F																								
G																								
H																								
I																								
J																								
K																								
Week number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**

[5]

(b) There are three teams working on the project. Each team is able to work on any of the activities.

Explain, with reference to the PERT chart, how work can be allocated to the three teams.

.....

.....

.....

.....

.....

..... [2]

(c) The PERT chart is used to calculate the critical path for the project.

(i) List the activities that form the critical path using the given PERT chart on page 12.

..... [1]

(ii) Explain the importance of the critical path for project delivery.

.....

.....

.....

.....

..... [2]

Question 52

6 A linked list abstract data type (ADT) is created. This is implemented as an array of records. The records are of type `ListElement`.

An example of a record of `ListElement` is shown in the following table.

Data Item	Value
`Country`	`"Scotland"`
`Pointer`	`1`

(a) (i) Use **pseudocode** to write a definition for the record type, `ListElement`.

.....

.....

.....

.....

..... [3]

(ii) Use **pseudocode** to write an array declaration to reserve space for only 15 nodes of type `ListElement` in an array, `CountryList`. The lower bound element is `1`.

..... [2]

(b) The program stores the position of the last node in the linked list in `LastNode`. The last node always has a `Pointer` value of `-1`. The position of the node at the head of the list is stored in `ListHead`.

After some processing, the array and variables are in the following state.

ListHead
1
LastNode
3

CountryList

	Country	Pointer
1	"Wales"	2
2	"Scotland"	4
3		-1
4	"England"	5
5	"Brazil"	6
6	"Canada"	7
7	"Mexico"	8
8	"Peru"	9
9	"China"	10
10		11
11		12
12		13
13		14
14		15
15		3

A **recursive** algorithm searches the list for a value, deletes that value, and updates the required pointers. When a node value is deleted, it is set to empty "" and the node is added to the end of the list.

A node value is deleted using the pseudocode statement

```
CALL DeleteNode("England", 1, 0)
```

Complete the following **pseudocode** to implement the `DeleteNode` procedure.

```
PROCEDURE DeleteNode(NodeValue: STRING, ThisPointer : INTEGER,
                                                PreviousPointer : INTEGER)

 IF CountryList[ThisPointer].Value = NodeValue

   THEN

     CountryList[ThisPointer].Value ← ""

     IF ListHead = ..........

       THEN

         ListHead ← ..........

       ELSE

         CountryList[PreviousPointer].Pointer ← CountryList[ThisPointer].Pointer

     ENDIF

     CountryList[LastNode].Pointer ← ..........

     LastNode ← ThisPointer

     ..........

   ELSE

     IF CountryList[ThisPointer].Pointer <> -1

       THEN

          CALL DeleteNode(NodeValue, ..........,
                                                              ThisPointer)

       ELSE

          OUTPUT "DOES NOT EXIST"

     ENDIF

 ENDIF

ENDPROCEDURE
```

[5]