Question No. 1

a) i)	+13 *mark as follows:* Exponent: +4 // move the pattern four places Mantissa: +13/16 // 0.1101 Answer: $13/16 \times 2^4$ // or equivalent	**3**
a) ii)	There will be a unique representation for a number. The format will ensure the number is represented with the greatest possible/more accuracy/precision. Multiplication is performed more accurately/precisely. [max 1]	**1**
a) iii)	Mantissa: 0100 0000 Exponent: 1000 Therefore number is $½ * 2^{-8}$ // +1/512 // $+2^{-9}$ // 0.00195	**3**
b)	The choices made will affect range and accuracy. More bits used for the mantissa will result in greater accuracy. More bits used for the exponent will result in a larger range of numbers.	**2**

Question No. 2

a) i)	00101000 00000011 =0.0101 × 2 ↑3 =10.1 =2.5	**3**
a) ii)	For a positive number (mantissa starts with a zero) bit after binary point (second bit from left) should be a one	**2**
a) iii)	00101000 00000011 = 01010000 00000010	**2**
b) i)	01111111 0111111	**2**
b) ii)	01000000 1000000	**2**
b) iii)	number will become too large to represent which will result in overflow	**2**
c)	Any point 1 mark 0.1 cannot be represented exactly in binary 0.1 represented here by a value just less than 0.1 the loop keeps adding this approximate value to counter until all accumulated small differences become significant enough to be seen	**3**

Question No. 3

a) i)	01101000 0011 = 0.1101 (or 1/2 + 1/4 + 1/16) × 2↑3 = 110.1 = 6.5	**3**
a) ii)	+3.5 = 11.1 = 0.111 × 2↑2 (or indication of moving binary point correctly) = 01110000 0010	**3**
a) iii)	01110000 Allow f.t. from **(ii)** 10001111 One's complement on mantissa 10001111 +1 Two's complement = 10010000 0010	**3**
b) i)	Precision/accuracy of numbers represented will increase	**1**
b) ii)	Range of numbers represented will increase	**1**
c)	Any point, 1 mark (max. 3) 0.1/0.2 cannot be represented exactly in binary // rounding error 0.1 represented by a value just greater than 0.1 // 0.2 represented by a value just greater than 0.2 adding two representations together adds the two differences summed difference significant enough to be seen	**3**

Question No. 4

a)	+2.5 = 010100000000 0010 Give full marks for correct answer (normalised or not normalised) = 10.1 = 0.101×2^2 // evidence of shifting binary point appropriately	**3**
b)	–2.5 101100000000 0010 Give full marks for correct answer One's complement of 12-bit mantissa of +2.5 101011111111 – allow f.t. +1 to get two's complement 101100000000	**3**
c)	3 Give full marks for correct answer = $0.011 \text{ X } 2^3$ // exponent is 3 = 11.0 // (1/4+1/8) * 8	**3**
d) i)	Not normalised	**1**
d) ii)	First two bits should be different for normalised number // because the number starts with 00	**1**
e)	reduced accuracy increased range	**2**

Question No. 5

a)	+3.5 01110000 00000010 Give full marks for correct answer (normalised or unnormalised) = 11.1 = 0.111×2^2 // evidence of shifting binary point appropriately	**3**
b)	–3.5 10010000 00000010 3 marks for correct answer One's complement of 8-bit mantissa for +3.5 10001111 – allow f.t. +1 to get two's complement 10010000	**3**
c)	14 3 marks for correct answer =0.111 X 2^4 // exponent is 4 =1110.0 / (1/2 + 1/4 + 1/8) * 16	**3**
d) i)	Normalised	**1**
d) ii)	Leftmost two bits are different for normalised representation // because the pattern starts with 01	**1**
e)	1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	**2**

Question No. 6

a)	**1** mark per bullet • 21.75 = 010101.11 (conversion to correct binary) • 0.1010111×2^5 (evidence of shifting binary point appropriately) • 01010111 0101 (stored as mantissa and exponent)	**3**

b)	**1** mark per bullet, max **2**
	
	• 1110 = –2 (conversion of exponent to denary)
	• 1.011000 = –0.101 (conversion of mantissa to negative binary number)// –0.625 (denary value of mantissa)// –5/8
	• –0.00101 (binary value) //
	
	Or
	• Use exponent to denormalise mantissa
	
	1 mark for correct answer
	• –5/32 // –0.15625
		3

Question No. 7

a)	**1** mark per bullet max **2** • 0101 = 5 (conversion of exponent to denary) • 1.01110011010 = –0.10001100110 (conversion of mantissa to negative binary number) • –10001.100110 (binary value)// –0.54980469 (denary value of mantissa) // –563/1024 Or • Use exponent to denormalise mantissa **1** mark for correct answer • = –17 19/32 // –17.59375	**3**
b)	**1** mark per bullet • 5.25 = 101.01 (conversion to binary) • = 0.10101 × 2^3 (evidence of shifting binary point appropriately) • 010101000000 0011 (stored as mantissa and exponent)	**3**
c)	**1** mark per bullet • (Size of mantissa decreased means that) precision is reduced • (Size of exponent is increased means that) range is increased	**2**

Question No. 8

a) i)	1 mark per bullet point: □ Correct value for exponent identified e.g. (0.0111 □ 2^)7 □ Used to give correct value e.g. 111 000 (1/4 + 1/8 +1/16) □ 128, 0.4375 □ Correct answer i.e. 56	**3**
a) ii)	The two most significant bits are 0 in the mantissa // In mantissa, 2nd bit is not the inverse of 1st bit	**1**
a) iii)	1 mark per bullet point: □ Mantissa = 01110000 □ Exponent = 0110	**2**
b) i)	1 mark per bullet point: □ Mantissa = 01111111 □ Exponent = 0111	**2**
b) ii)	1 mark per bullet point: □ Mantissa = 01000000 □ Exponent = 1000	**2**
c) i)	Precision of numbers represented will increase	**1**
c) ii)	Range of numbers represented will increase	**1**
d)	1 mark per bullet point to max 3: □ 0.1/0.2/0.3 cannot be represented exactly in binary / rounding errors □ adding two or more inaccurate representations together increases the probability of inaccuracy □ giving an answer where the difference is significant enough to be seen	**3**

Question No. 9

a) i)	1 mark per bullet point: • Correct value for exponent identified e.g. (0.010101 × 2^)5 • Used to give correct value e.g. 1010.1 or 21/64 x 32 • Correct answer i.e. 10.5 // 10½	**3**
a) ii)	1 mark per bullet point: • Correct binary value i.e. 111.1 • Value for exponent identified e.g. (0.1111 × 2^)3 • Correct answer i.e. 01111000 00000011	**3**
a) iii)	1 mark per bullet point: • Any working method for conversion • Applied accurately • Correct answer i.e. 10001000 00000011	**3**
b) i)	Largest (positive) number (in this format)	**1**
b) ii)	Overflow // too large to represent // would become negative	**1**

Question No. 10

a) i)	**2 marks** for working **1 mark** for correct answer Working: □ = 0. 0110111 x 2^5 places // exponent = 5 □ = 1101.11 (moving bp 5) Answer: □ = 13.75 // 13 ¾	**3**
a) ii)	The first two bits of the mantissa are 0 / the same / not different / are not 01	**1**
a) iii)	**1 mark** per bullet point □ Mantissa = 01101110 □ Exponent = 0100	**2**
b) i)	**2 marks** for working **1 mark** for correct answer Working: □ 01011.101 □ 0.1011101 □ 2^4 // showing calculation of exponent = 4 Answer: □ 01011101 0100	**3**
b) ii)	**2 marks** for working **1 mark** for correct answer Working: □ 10100.011 // 10100011 correct use of two's complement or other method □ Exponent = 4 Answer: □ 10100011 0100	**3**
c)	**1 mark** per bullet point (max 3) □ 0.2/0.4 cannot be represented exactly in binary / rounding error □ 0.2 has been represented by a value just greater than 0.2 // 0.4 has been represented by a value just greater than 0.4 □ Therefore multiplying these two representations together increases the difference □ difference after the calculation is significant enough to be seen (given the number of positions after the decimal place)	**3**

Question No. 11

a)	**2 marks** for working shown **1 mark** for the correct answer Working: • Correct calculation of negative value (any method) (= –0.11010001101) • Correctly moving the binary point 7 places (= –01101000.1101) // Exponent 7 Answer: • -104.8125 // $-104\frac{13}{16}$	**3**
b)	**2 marks** for working shown **1 mark** for the correct answer Working: • Correct conversion to binary (01.1001) • Correct calculation of exponent (1) Answer: • (Mantissa) 0110 0100 0000 (Exponent) 0001	**3**
c) i)	**1 mark** per bullet point • Mantissa = 0111 1111 1111 • Exponent = 0111	**2**
c) ii)	**1 mark** per bullet point • Mantissa = 0100 0000 0000 • Exponent = 1000	**2**
d)	**1 mark** per bullet point to **max 3** • The trade-off is between range and precision • Any increase in the number of bits for the mantissa, means fewer bits available for the exponent // Any decrease in the number of bits for the mantissa, means more bits available for the exponent • More bits used for the mantissa will result in better precision • More bits used for the exponent will result in a larger range of numbers • Fewer bits used for the mantissa will result in worse precision • Fewer bits used for the exponent will result in a smaller range of numbers	**3**

Question No. 12

a) i)	**1 mark** per bullet point • Exponent 0010 = 2 • Mantissa 0.1010010 becomes 010.10010 // $\frac{41}{64}$ // $2+\frac{1}{2}+\frac{1}{16}$ • Answer $2\frac{9}{16}$ // 2.5625	**3**
a) ii)	**1 mark** per bullet point • $-3.75 = 100.01000$ // $-4+\frac{1}{4}$ / 0.25 • 100.01000 becomes 1.0001000 Exponent = +2 • Answer: Mantissa = 10001000 Exponent = 0010	**3**
b)	Only the range is increased (no effect on precision)	**1**
c)	**1 mark** per bullet point to **max 1** • There is no **exact** binary conversion for some numbers • More bits are needed to store the number than are available	**1**
d)	First term: Overflow Second term: Underflow	**2**

Question No. 13

a) i)	1101	**1**
a) ii)	011100000000	**1**
a) iii)	1 mark for positive, 1 for justification • Positive … • … the most significant / first bit in the mantissa is 0	**2**
a) iv)	1 mark per bullet point • Exponent = 1011 = −3 // binary point moved 3 places left • Mantissa 0.111 becomes 0.000111 // ⅞ // ½ + ¼ + ⅛ // $2^{-1} + 2^{-2} + 2^{-3}$ • Answer: 7 / 64 // 0.109375	**3**
b)	1 mark per bullet point • Increases the range • Decreases the precision	**2**

Question No. 14

a)	= (0)11000000.1 (conversion to binary) [1] = 0.110000001×2^8 (evidence of shifting binary point appropriately) [1] = 0110000001 001000 (stored as mantissa and exponent) [1]	**3**
b)	1001111110 (one's complement of 10 bit mantissa) [1] 1001111111 (two's complement of 10 bit mantissa) [1] 1001111111 001000 (stored as mantissa and exponent) [1]	**3**
c)	Any **three** from: • Exponent too large to fit in 4 bits as a two's complement number • Exponent will turn negative/–8 • … therefore, point moves the wrong way • Value will be approx. +0.0029(296875)	**3**

Question No. 15

a)	Exponent = 5 (conversion of exponent to denary) 0.00011 or 0.09375 or 3/32 (value of mantissa) //moving of binary point 3 (answer)	**3**
b) i)	Mantissa = 011000000000 Exponent = 0010	**2**
b) ii)	Any **two** from Precision lost Redundant leading zeros in the mantissa Bits lost off right hand end / least significant end Multiple representations of a single number	**2**

Question No. 16

a)	Exponent = 6 (conversion of exponent to denary) 0.101 or 0.625 or 5/8 (value of mantissa) // moving of binary point 40 (answer)	**3**
b) i)	Exponent = 5 (conversion of exponent to denary) 0.00000000110 or 3/1024 (value of mantissa) // moving of binary point 0.09375 or 3/32 (answer)	**3**
b) ii)	Any **two** from The number calculated will change The same bit pattern is for a different number Software may crash (if not updated)	**2**

Question No. 17

a) i)	A – negative, mantissa starts with a one B – positive mantissa starts with a zero	**3**
a) ii)	A mantissa: −0.5 // $-\frac{1}{2}$ A exponent: 2 B mantissa: 0.875 // 7/8 B exponent: −1	**4**
a) iii)	A: −2 B: 0.4375 // 7/16	**2**
b)	Number: **B** Justification: Using the mantissa The first two bits are different // first bit 0 second bit 1	**3**

Question No. 18

a) i)	A mantissa: 0.75 // 3/4 A exponent: −1 B mantissa: −0.25 // –1/4 B exponent: 4	**4**
a) ii)	A: 0.375 // 3/8 B: −4	**2**
b)	Number **A** Justification **Using** the mantissa The first two bits are different // first bit 0 second bit 1	**3**

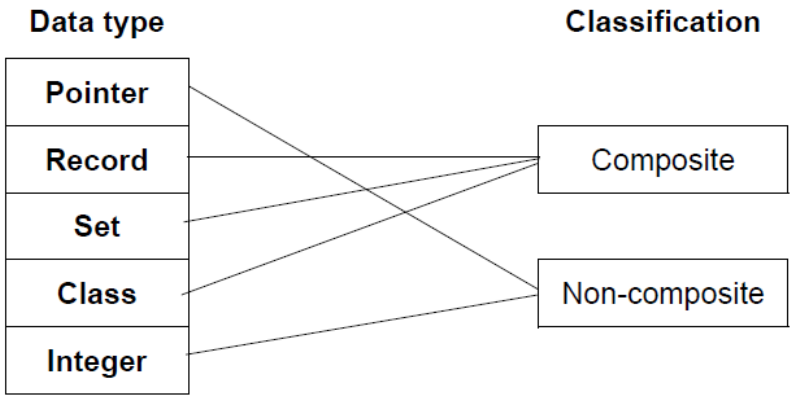
Question No. 19

a)	Working: **one mark** for calculation of the mantissa and **one mark** for calculation or use of the exponent **Exponent: one** from: = 0.11101×2^3 // 0.11101×2^{11} // 0.11101×10^3 // 0.11101×10^{11} = 1.00011×2^3 // 1.00011×2^{11} // 1.00011×10^3 // 1.00011×10^{11} = appropriate shifting of binary point for +7.25 **Mantissa: one** from: = 111.01 (conversion to binary +7.25 – 10 bits) = 0111010000 (mantissa 10 bits for +7.25 = 1000101111(one's complement mantissa for –7.25) = 1000110000 (two's complement mantissa for –7.25) Correct Answer **(Max 1)** Mantissa: 1 0 0 0 1 1 0 0 0 0 — Exponent: 0 0 0 0 1 1	**3**
b)	**One** mark for working out the exponent **One** mark for working out the mantissa **One** mark for the correct answer Example answers • =1.011000111×2^7 (exponent is 7) • =10110001.11 // –128 + 32 + 16 + 1 + 0.5 + 0.25 // convert to positive 01001110.01 (and add a minus sign to the answer) • –78.25	**3**
c)	**One** mark for working **One** mark for correct mantissa **One** mark for correct exponent Example answers Number of places added to exponent for normalisation −6 for number to retain its value // mantissa moved 6 places left Mantissa: 0 1 1 1 0 0 0 0 0 0 Exponent: 1 0 0 0 0 1	**3**
d) i)	**One** mark for each correct marking point (**Max 3**) • Requires 11 bits / more than 10 bits to store (accurately) / reference to maximum (positive) number that can be stored = 511 • Denary 513 in binary is 1000000001 // Normalised: 0.1000000001 • Results in overflow	**3**
d) ii)	**One** mark for each correct marking point (**Max 2)** • The number of bits for the mantissa must be increased • 11/12 bits mantissa **and** 5/4 bits exponent	**2**

Question No. 20

a) i)	**One** mark for each correct marking point (**Max 2)** • 010111000110 (correct mantissa) • 0111 (correct exponent)	**2**
a) ii)	**One** mark for each correct consequence **One** mark for each correct justification **Consequence** • The precision/accuracy of the number would be reduced **Justification** • … because the least significant bits of the original number have been truncated/lost // the original number had 13 bits / 14 bits with sign but the mantissa can only store 12 bits	**2**
b)	**One** mark for each correct marking point (**Max 3)** • To store the maximum range of numbers in the minimum number of bytes / bits • Normalisation minimises the number of leading zeros/ones represented • Maximising the number of significant bits // maximising the (potential) precision / accuracy of the number for the given number of bits • … enables very large / small numbers to be stored with accuracy. • Avoids the possibility of many numbers having multiple representations.	**3**

Question No. 21

a)	**2** marks for all 5 single lines correct **1** mark for 4 lines correct otherwise zero  	**2**
b) i)	Type **Enumerated** Classification **Non-composite**	**2**
b) ii)	`DECLARE session : timeOfDay` `session ← afternoon`	**2**

Question No. 22

a) i)	Composite box Non-composite size / enumerated REAL STRING	**4**
a) ii)	size	**1**
b)	myBox[1].volume ← medium myBox[1].price ← 10.99 myBox[1].colour ← "red"	**3**

Question No. 23

a)	**1 mark** per bullet point to **max 2** • No suitable data type is provided by the language used • The programmer needs specify a new data type • … that meets the requirements of the application / program	**2**

b) i)	**1 mark** per bullet point
	
	• `EmployeeID` declared as `STRING`
	• `Sales, Technical` and `CustomerServices` …
	• … with commas in-between
	• `ENDTYPE`
	
	`TYPE Employee`
	`  DECLARE EmployeeID : STRING`
	`  DECLARE EmployeeName : STRING`
	`  DECLARE Department : (Sales, Technical, CustomerServices)`
	`  DECLARE Salary : 25000..150000`
	`ENDTYPE`
		4

b) ii)	`DECLARE NewEmployee : Employee`	**1**
b) iii)	`NewEmployee.EmployeeID ← "02244"`	**1**
b) iv)	**1 mark** per bullet point to **max 2** • Array • List • Set • Collection • Class • Stack • Queue • Linked list • Dictionary	**2**

Question No. 24

a)	1 mark per bullet point to **max 2** • Derived from one or more existing data types • Used to extend the built-in data types • Creates data-types specific to applications // programmer's requirements	**2**
b) i)	Enumerated (data type)	**1**
b) ii)	`DECLARE CurrentMonth : Months`	**1**
b) iii)	`CurrentMonth ← August`	**1**

Question No. 25

a) i)	enumerated	**1**
a) ii)	record	**1**
a) iii)	`MyMonthOfBirth ← DateOfBirth.ThisMonth`	**1**

b) i)	`TYPE LocationRainfall`
	`  DECLARE LocationName : STRING`
	`  DECLARE LocationHeight : INTEGER`
	`  DECLARE TotalMonthlyRainfall : ARRAY[1..12] OF REAL`
	`ENDTYPE`
		5

b) ii)	• no need to re-sort data every time new data is added • only a small file so searching will require little processing • new records can easily be appended	**2**

Question No. 26

a) i)	`HomeAddress.ThisHouseNo ← 34`	**1**

a) ii)	`DECLARE ThisHouseNo: 1..10`
	
	`DECLARE ThisTown: [Brightown, Arunde, Shoram]`
		2

b) i)	`TYPE WeatherStation`
	`  DECLARE StationID : STRING`
	`  DECLARE Latitude : REAL`
	`  DECLARE Temperature : ARRAY[1..15] OF INTEGER`
	`ENDTYPE`
		5

b) ii)	`StationID` is hashed to produce home location If home location is free insert record Else use overflow method to find free location	**3**

Question No. 27

a) i)	`DECLARE NewFriend : MyContactDetail`	**1**
a) ii)	`NewFriend.HouseNumber ← 129`	**1**

b)	Declaration of `Name, Area, HouseNumber` 1
	Inclusion of three correct values for `Area` 1
	Inclusion of correct range for `HouseNumber` 1
	
	For example:
	
	`TYPE MyContactDetail`
	`  DECLARE Name : STRING`
	`  DECLARE Area : (uptown, downtown, midtown)` **1**
	`  DECLARE HouseNumber : 1..499` **1 & 1**
	`ENDTYPE`
		3

c) i)	4402	**1**
c) ii)	33	**1**
c) iii)	3427	**1**
c) iv)	`TRUE`	**1**
d) i)	`IPointer ← @MyInt2`	**1**
d) ii)	`MyInt1 ← 33`	**1**
d) iii)	`IPointer^ ← MyInt2`	**1**

Question No. 28

a) i)	`DECLARE Book : LibraryBookRecord`	**1**
a) ii)	`Book.Title ← "Dune"`	**1**

b)	`TYPE LibraryBookRecord`
	`  DECLARE ISBN : INTEGER`
	`  DECLARE Title : STRING`
	`  DECLARE Genre : (Fiction, Non-Fiction)` **1**
	`  DECLARE NumberOfLoans : 1 .. 99` **1**
	`ENDTYPE`
	mark for correct declaration and first two fields (**note**: only if attempt at modification) **1**
		3

c) i)	6715	**1**
c) ii)	8216	**1**
c) iii)	88	**1**
c) iv)	`FALSE`	**1**
d) i)	`Temp2 ← 22`	**1**
d) ii)	`IntPointer ← @Temp1`	**1**
d) iii)	`IntPointer^ ← Temp2`	**1**

Question No. 29

a)	`CollegeStudent.StudentID ← 6539`	**1**
b) i)	**1** mark per bullet • `StudentCourse: ARRAY[1:6] OF` • All valid string options , for example: `DECLARE StudentCourse: ARRAY[1:6] OF ("Computer Science", "Engineering", "Science", "Maths", "Physics", "Chemistry", "Music", "Drama", "English Language")`	**2**
b) ii)	`DECLARE StudentID: 1 .. 8000`	**1**

c) i)	**1** mark per bullet
	• Type declaration `TYPE` and `ENDTYPE`
	• Declaring `Code` as `STRING`
	• Declaring `Mark` as `ARRAY [1:6] OF INTEGER`
	• `AverageMark` as `REAL`
	For example:
	`TYPE StudentAssessment`
	`  DECLARE Code : STRING`
	`  DECLARE Mark : ARRAY[1:6] OF INTEGER`
	`  DECLARE AverageMark : REAL`
	`ENDTYPE`
		4

c) ii)	Any **3** from, **1** mark per bullet • `StudentID`/key field is hashed to produce home location • If home location is free, insert record/data • Else use overflow method to find free location to store record / data • If no free location available then file is full and record/data cannot be stored	**3**

Question No. 30

a)	single data type that does not involve a reference to another type/usually built in to a programming language	**1**
b)	**1** mark for data type, **1** for definition, max **4**, 2 data types • Integer • Stores a whole number • Boolean • Stores true or false/1 or 0/on or off • Real/Single/Double/Float/Decimal • Stores decimal numbers • String • Stores zero or more characters • Char • Stores a single character • Pointer • Whole number used to reference a memory location	**4**
c)	data type constructed from other data types	**1**
d)	**1** mark for naming, **1** for description, max **4**, 2 data types • Record • collection of related items which may have different data types • Array • (Indexed) collection of items with the same data type • List • (Indexed) collection of items that can have different data types • Set • stores a finite number of different values that have no order // supports mathematical operations • Class/Structure • Gives the properties and methods for an object	**4**

Question No. 31

a)	Record	**1**
b)	Enumerated	**1**
c)	`DECLARE BestSeller : Book`	**1**
d)	`BestSeller.Author ← "John Williams"`	**1**

Question No. 32

a)	**One** mark for each correct marking point **(Max 2)** • To create a new data type (from existing data types) • To allow data types not available in a programming language to be constructed // To extend the flexibility of the programming language	**2**
b) i)	`TYPE SchoolDay = (Monday, Tuesday, Wednesday, Thursday, Friday)`	**1**
b) ii)	`TYPE WeekEnd = (Saturday, Sunday)`	**1**

c)	**One** mark for each marking point (**Max 4**)
	
	• `TYPE ClubMeet` and `ENDTYPE` correct
	• `DECLARE FirstName` and `DECLARE LastName` included with correct data types
	• `DECLARE Schoolday` included with correct data types from part 2(b)(i)
	• `DECLARE Weekend` included with correct data types from part 2(b)(ii)
	
	Example answer
	
	`TYPE ClubMeet`
	`  DECLARE FirstName : STRING`
	`  DECLARE LastName : STRING`
	`  DECLARE Schoolday : SchoolDay`
	`  DECLARE Weekend : WeekEnd`
	`ENDTYPE`
		4

Question No. 33

a)	**One** mark for each marking point (**Max 2**) • `TYPE Parts =` • `(Monitor, CPU, SSD, HDD, LaserPrinter, Keyboard, Mouse)` **Complete answer** `TYPE Parts = (Monitor, CPU, SSD, HDD, LaserPrinter, Keyboard, Mouse)`	**2**
b)	**One** mark for each marking point (**Max 2**) • `TYPE SelectParts = ^` • correct data type chosen `Parts` **Complete answer** `TYPE SelectParts = ^Parts`	**2**

Question No. 34

a)	`SchoolComputer.ComputerID ← 1234` `SchoolComputer.ComputerLocation ← Lab2`	**2**
b) i)	`DECLARE StudentID : ARRAY[1:20] OF INTEGER`	**1**
b) ii)	`DECLARE ComputerID : 1000 .. 1999 //` `DECLARE ComputerID : INTEGER 1000 .. 1999`	**1**
c)	Any **three** from • Computer ID hashed to give address / home location • Compared to ID stored at address / home location • Nothing stored, output message 'record not found' • Record IDs equal, record is found • Record IDs not equal, search overflow area / next record • Until record found or whole area searched • If no record found error message	**3**

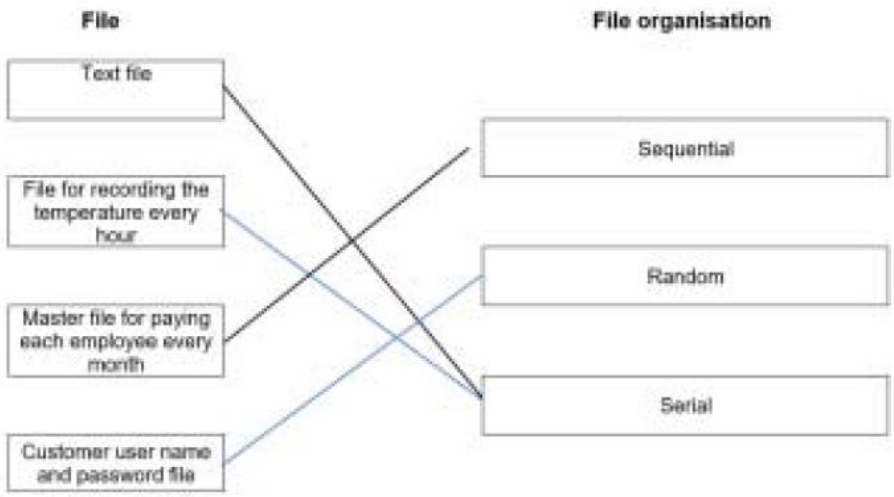
Question No. 35

a)	`LoanBicycle.BicycleID ← 567` `LoanBicycle.BicycleAvailable ← FALSE`	**2**
b) i)	`DECLARE BorrowerID : ARRAY[1:10] OF INTEGER`	**1**
b) ii)	`DECLARE BicycleID : 500 .. 599 //` `DECLARE BicycleID : INTEGER 500 .. 599`	**1**
c)	Any **three** from • Bicycle ID **hashed** to give address / home location • Check if a record already stored at address / home location • If nothing stored, store new record • If another record already stored search overflow area / next record • Until free space found or whole area searched • If no space output error message	**3**

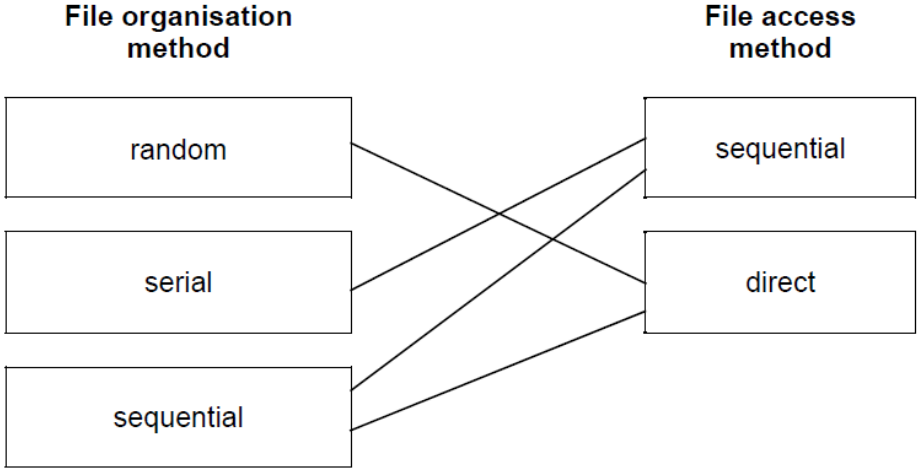
Question No. 36

a)	**One** mark for each correct marking point **(Max 4)** • In both serial and sequential files records are stored one after the other … • … and need to be accessed one after the other • Serial files are stored in chronological order • Sequential files are stored with ordered records • … and stored in the order of the key field • In serial files, new records are added in the next available space / records are appended to the file • In sequential files, new records are inserted in the correct position.	**4**
b)	Direct (access)	**1**
c)	Sequential (access)	**1**

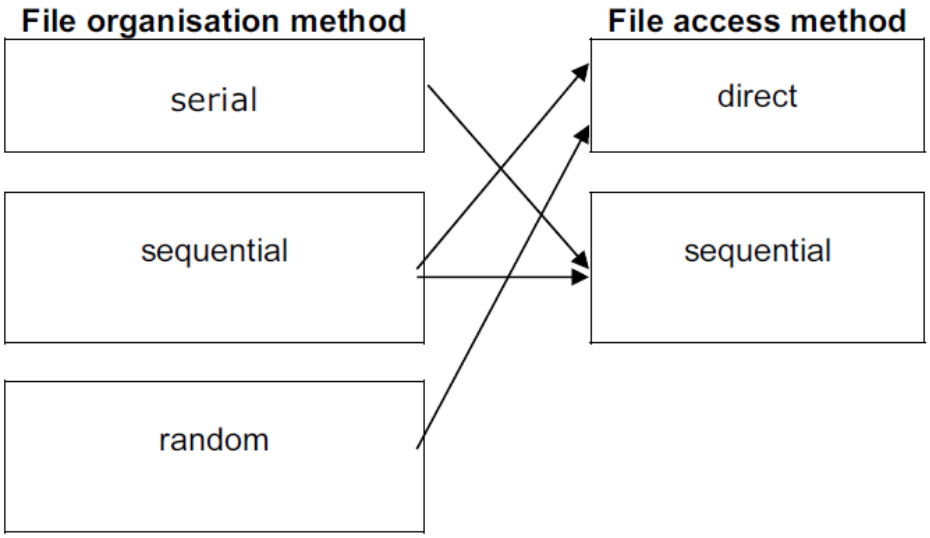
Question No. 37

-	**One** mark for each correct line drawn  	4

Question No. 38

a)	 1 mark for random correct 1 mark for serial correct 2 marks for sequential correct (1 per correct line)	**4**
b) i)	File A: Serial 1 Meter readings are submitted over time // added to the end of file 1 Stored chronologically 1	**3**
b) ii)	File B: Sequential 1 Any two points from: Each customer has a unique account number 1 Sorted on Account number 1 High hit rate // Suitable for batch processing monthly statements 1	**3**
b) iii)	File C: Random 1 Login without waiting // Random organisation allows fastest direct access to required record 1 Low hit rate // Suitable for access to individual records 1	**3**

Question No. 39

a)		**4**
b) i)	Sequential As all customers get statement … // high hit rate Suitable for batch processing of the records // the records will be processed one after the other File organised using customer's unique ID (as primary key field) // Serial As all customers get statement … // high hit rate Suitable for batch processing of the records // the records will be processed one after the other Order not important	**3**
b) ii)	Random Real-time transaction processing Requires fastest access to data No need to search through records	**3**
b) iii)	Serial Each new record is appended Transactions are recorded in chronological order File re-organisation not required for each new record // no need for the records to be sorted	**3**

Question No. 40

a) — **1**

Example:
Speed of access
Just used as a look-up file
No need for any serial or sequential processing
1 mark for any valid point

b) i) — **1**

CustomerID	RecordKey
802139	2139
700004	4
689998	89998
102139	2139

b) ii) — **2**

Minimum value: 0 1
Maximum value: 99999 1

b) iii) — **4**

```
PROCEDURE InsertRecord(CustomerID : INTEGER)
   RecordKey ← CustomerID MOD 100000
   Success ← FALSE
   // Find position for new record and insert it
   REPEAT
      IF record at position RecordKey is empty
         THEN
            Insert new record at position RecordKey
            Success ← TRUE
         ELSE
            IF RecordKey = 99999
               THEN
                  RecordKey ← 0
               ELSE
                  RecordKey ← RecordKey + 1
            ENDIF
      ENDIF
   UNTIL Success = TRUE
ENDPROCEDURE
```

c) i) — **2**

For security
If file is hacked then encrypted PIN cannot be used
Only encrypted PINs are transmitted and compared
1 mark for any valid point

c) ii) — **3**

1. Customer ID is read from card
2. Customer enters PIN
3. Customer PIN is **encrypted**
4. **Customer ID is hashed**
5. Customer record is located in file
6. **PIN is checked against PIN in record**
7. If match then transaction can proceed