

ISC Class 12 Computer Science Specimen Paper 2

Written by the Future Topper subject team on the CISCE examination pattern. Not an official CISCE paper.
Found a mistake? Write to contactfuturetopper@gmail.com

Maximum Marks: 70 | Time Allowed: Three hours | Reading Time: 15 minutes

Paper 1 (Theory) carries 70 marks. Part I is compulsory. In Part II attempt two questions from Section A, two from Section B and one from Section C.

General Instructions

Write your answers on the separate answer paper provided. The first 15 minutes are for reading the paper, and you may not write during that time. Part I is compulsory and carries 20 marks. Part II carries 50 marks and is divided into three Sections. From Section A answer any two of the three questions, from Section B answer any two of the three questions, and from Section C answer any one of the two questions. Marks for each question, or part of a question, are shown in brackets. The programming language is Java, and all programs and program fragments must be written in Java. Give meaningful names to your variables, and indent your code so that the structure of every block is clear. No circuit diagrams, Karnaugh maps, truth tables, topology sketches or linked list figures are printed in this paper. Where a question asks for one you are expected to draw it yourself in pencil and label it. This paper carries 70 of the 100 marks for the subject, the remaining 30 being for the practical work of Paper 2.

PART I (20 Marks)

Answer all questions. Question 1 carries 10 marks and Question 2 carries 10 marks.

Question 1 [10]

Each part carries 1 mark. For the parts that offer four choices, write only the option you choose. For the rest, answer in a single sentence or give the single word, name, value or expression asked for.

- (i) State the absorption law of Boolean algebra in the form that reduces $A + AB$, and give the reduced expression.
- (ii) The decimal value of the hexadecimal number $(2F5)_{16}$ is:
(a) 629 (b) 757 (c) 773 (d) 885
- (iii) The gate that is called universal because every other gate can be built from copies of it alone is:
(a) XOR (b) AND (c) NOT (d) NAND
- (iv) Name the bus of a computer that carries the address of the memory location being read or written, and state why it carries signals in one direction only.
- (v) The number of address lines a processor needs in order to reach every byte of a memory of 64 kilobytes is:
(a) 8 (b) 32 (c) 16 (d) 64
- (vi) State what the keyword `super` does when it appears as the first statement inside a constructor.
- (vii) A variable declared inside an interface is by its nature:
(a) `public`, `static` and `final` (b) `private` and `static`
(c) `protected` and `final` (d) `public` and `transient`
- (viii) Name the data structure in which an element may be added and removed at one end only, and state the three letter name for the order in which its elements leave.
- (ix) The topology in which every node is joined directly to every other node, so that a break in any one link leaves all the nodes still reachable, is:
(a) bus (b) star (c) ring (d) mesh

- (x) The act of sending a message that appears to come from a bank, in order to persuade the reader to type a password into a counterfeit page, is called:
- (a) phishing (b) a denial of service attack
(c) plagiarism (d) data mining

Question 2 [10]

Each part carries 2 marks.

- (i) Convert the binary number $(10110110)_2$ into its octal equivalent and into its hexadecimal equivalent, showing how the bits are grouped in each case. [2]
- (ii) Give the output of the following fragment, and state in one sentence why the first printed value begins with a digit. [2]
- ```
StringBuilder sb = new StringBuilder("INTERFACE");
sb.delete(2, 5);
sb.insert(0, sb.length());
System.out.println(sb);
System.out.println(sb.reverse());
```
- (iii) Add the binary numbers  $(1011)_2$  and  $(1101)_2$ , showing the carry generated in each column, and check your answer by converting all three numbers into decimal. [2]
- (iv) State two differences between method overloading and method overriding in Java. [2]
- (v) A linear queue is held in an array of five places, with front and rear both starting at 0 and minus 1. After six values have been added and two removed, the queue reports itself full although two places are free. Explain why this happens, and name the change to the queue that cures it. [2]

## PART II (50 Marks)

Answer two questions from Section A, two questions from Section B and one question from Section C.

### SECTION A

Answer any two questions from this Section. Each question carries 10 marks.

#### Question 3 [10]

- (a) State both of De Morgan's theorems in symbols, and prove the first of them by drawing a truth table for yourself. The table should carry columns for the two inputs A and B, for the sum  $A + B$ , for its complement and for the product of the two complements, and all four rows must be filled in. [4]
- (b) Reduce the Boolean expression  $(A + B)(A + B')(A' + C)$  to its simplest form using Boolean laws alone, naming the law used at each step. State how many literals the reduced form contains. [3]
- (c) Carry out the subtraction 71 minus 45 in eight bit binary using the 2's complement method. Show the eight bit form of each number, the 2's complement of the number being subtracted, the addition and what becomes of the carry out of the leftmost column. Then state how the method signals that an answer is negative, using 25 minus 58 as your illustration and giving its magnitude in decimal. [3]

#### Question 4 [10]

- (a) A function F of the four variables A, B, C and D is 1 for the input combinations numbered 0, 1, 4, 5, 6, 7, 9 and 13, counting the combinations from 0 to 15 with A as the most significant variable. Reduce F to its simplest sum of products form using a Karnaugh map for four variables. Draw the map yourself, using Gray code ordering along both edges, mark every square that holds a 1, and ring each group you use, stating the size of each group and the term it yields. [5]
- (b) A second function G of the same four variables is 0 for the input combinations numbered 2, 3, 6, 7, 8, 9, 12 and 13, and is 1 for every other combination. Reduce G to its simplest product of sums form using a

Karnaugh map of zeros. Draw the map, ring each group of zeros, state the sum term each group yields, and name the single two input gate that performs the function you obtain. [5]

### Question 5 [10]

- (a) Explain what a full adder does and name its three inputs. Draw its truth table yourself, with columns for the three inputs and for the two outputs, sum and carry out, and fill in all eight rows. Write the Boolean expression for the sum output as an exclusive OR of the three inputs. [4]
- (b) Distinguish between primary and secondary memory under three headings, namely volatility, the way the processor reaches the contents and the usual relative cost for each byte stored. Then explain why a computer is fitted with cache memory when it already has primary memory. [3]
- (c) A company has been given the network address 192.168.10.0 with the mask 255.255.255.192. State the number of host bits this mask leaves, the total number of addresses in the block, the number of those that can be given to machines and the broadcast address of the block. Then name the protocol that hands out addresses to machines automatically as they join a network. [3]

## SECTION B

Answer any two questions from this Section. Each question carries 10 marks. In this Section write your answers in Java.

### Question 6 [10]

- (a) Define a class named Hostel as described below, and write a main method that uses it. [10]

Class name: Hostel

Data members: student of type String to hold the name of the resident, units of type int to hold the units of electricity consumed, and bill of type double to hold the amount payable. All three are to be private.

Hostel(String s, int u): a parameterised constructor to set the name and the units, and to set the bill to zero  
void computeBill(): to work out the bill on these rules. For the first 100 units the rate is 4.0 rupees a unit.

For units beyond 100 and up to 300 the charge is 400.0 rupees for the first hundred together with 6.5 rupees for every unit above 100. Beyond 300 units the charge is 1700.0 rupees for the first three hundred together with 8.0 rupees for every unit above 300. After the charge has been worked out, a surcharge of 5 per cent of it is to be added if, and only if, the charge exceeds 1500.0 rupees.

void display(): to print the name, the units and the bill, each on a line of its own, in that order

Write a main method that creates three objects, one for Riya Sharma with 80 units, one for Imran Qureshi with 250 units and one for Tara Nair with 400 units, and calls computeBill and display on each of them in turn. State the complete output the program produces.

### Question 7 [10]

Give the output of each of the following three fragments. Show the working wherever a value changes more than once.

- (a) Give the output of the following fragment, which is the body of main, and state the number of times the loop body is executed. [3]

```
int p = 1, q = 9;
while (p < q)
{
 p = p * 2;
 q = q - p / 2;
 System.out.print("(" + p + "," + q + ") ");
}
System.out.println();
```

```
System.out.println(p + q);
```

- (b)** Give the output of the following program, and explain in one sentence why the first call to measure gives the answer it does although the variable was declared as being of the parent type. [4]

```
class Shape
{
 protected int side;
 Shape(int s)
 {
 side = s;
 System.out.println("Shape built with side " + side);
 }
 int measure()
 {
 return 4 * side;
 }
}
class Cube extends Shape
{
 Cube(int s)
 {
 super(s);
 System.out.println("Cube built");
 }
 int measure()
 {
 return 6 * side * side;
 }
 int baseMeasure()
 {
 return super.measure();
 }
}
public class Solid
{
 public static void main(String args[])
 {
 Shape ob = new Cube(3);
 System.out.println(ob.measure());
 Cube cb = new Cube(2);
 System.out.println(cb.baseMeasure());
 }
}
```

- (c) Give the output of the following two recursive methods for the calls shown, and state what each method computes. [3]

```
static int g(int n)
{
 if (n == 0)
 return 0;
 return n % 10 + g(n / 10);
}
static String h(int n)
{
 if (n == 0)
 return "";
 return h(n / 2) + (n % 2);
}
public static void main(String args[])
{
 System.out.println(g(4729));
 System.out.println(h(45));
}
```

### Question 8 [10]

- (a) Write a complete Java program that reads eight single word names typed by the user into an array of String, arranges them in alphabetical order using the bubble sort technique only, prints the sorted names one to a line, and then reads one further name and reports whether it is in the list, giving its position counted from one if it is. Compare the names with compareTo and equals rather than with the relational operators, and do not use any ready made sorting or searching method. State the output your program gives for the eight names Vikram, Anita, Suresh, Bhavna, Nitin, Deepa, Kavya and Rahul when the name searched for is Nitin. [6]
- (b) Write a complete Java program that writes two given lines of text into a file named notes.txt, then reads that file back line by line and prints the number of lines, the total number of words and the number of words longer than five letters. Take the two lines as Data travels as packets and Every packet carries an address, treat a single space as the separator between words, and put the whole of the file handling inside a try block with a catch for an input or output failure. State the complete output. [4]

## SECTION C

Answer any one question from this Section. The question carries 10 marks.

### Question 9 [10]

- (a) Define a class named Ring that holds a circular queue of integers in an array, as described below. [7]
- Data members: q, an array of int to hold the elements, front, an int to hold the index of the element at the front, rear, an int to hold the index of the element at the rear, count, an int to hold the number of elements present, and cap, an int to hold the capacity. All five are to be private.
- Ring(int n): a parameterised constructor that creates the array with n places, stores the capacity, sets front to 0, sets rear to minus 1 and sets count to 0
- void addRear(int v): to add v at the rear if the queue is not full, moving rear round to the start of the array when it passes the last place, and otherwise to print a message that reports the queue as full and names the value that could not be added

int removeFront(): to remove and return the element at the front, moving front round in the same way, and, if the queue is empty, to print a message saying so and to return minus 1

Write a main method that creates a ring of capacity four, adds 7, 14 and 21, removes and prints two values, then adds 28, 35, 42 and 49 in that order, and finally removes and prints five values. State the complete output, and explain in one sentence what the remainder operator is doing to front and rear.

- (b) A singly linked list of integers is held with a reference named head. Explain why counting the nodes of such a list takes a number of steps that grows with the length of the list, while reading the element at a given index of an array takes the same time whatever the index. Then state one advantage a linked list has over an array in return. [3]

#### Question 10 [10]

- (a) Explain what is meant by intellectual property in software, and distinguish between copyright, a patent and a trademark by stating what each one protects in a piece of software. Then explain what free and open source software is, and state why copying a few paragraphs of another person's blog into your own project report without naming the source is wrong even where no money has changed hands. [5]
- (b) Explain what big data is, and state three properties by which it is usually characterised. Then explain in one sentence why holding a very large quantity of personal data is a risk to the person who holds it as well as to the people it describes. [3]
- (c) Explain what a blockchain is, and state the property of it that makes an entry once recorded very hard to alter quietly. [2]

## ANSWER KEY AND MARKING SCHEME

Full code is given for every question that asked for a program or a class, and every program in this key has been compiled and run, so the outputs quoted are the outputs actually produced. Marks shown in italics are the step marks an examiner would award. Accept any correct alternative logic that produces the same output, accept any sensible variable names, and do not deduct marks for a different but consistent style of indentation or bracing. For an output question the printed text must match exactly, including spaces. For a Boolean reduction accept any form that is logically equivalent to the one given and uses no more terms. For a drawing accept any clear and correctly labelled version.

### Question 1 [10]

- (i) The absorption law states that  $A + AB = A$ . The proof in one line is  $A + AB = A(1 + B) = A.1 = A$ , since  $1 + B = 1$  whatever  $B$  is. The reduced expression is therefore  $A$ , and the term  $AB$  is swallowed by  $A$  because every case in which  $AB$  is 1 is already a case in which  $A$  is 1.
- (ii) (b) 757. Working:  $2 \times 256 + 15 \times 16 + 5 = 512 + 240 + 5 = 757$ .
- (iii) (d) NAND. A NAND gate with both inputs tied together gives NOT, two NAND gates in series give AND, and inverting both inputs before a NAND gives OR, so any circuit at all can be built from NAND gates alone. NOR is universal in the same way, but NOR is not among the options offered.
- (iv) The address bus. It carries signals in one direction only, out from the processor to the memory and the input and output units, because only the processor ever places an address on it, the addressed unit having no occasion to send an address back. The data bus, by contrast, must be bidirectional because data travels both ways.
- (v) (c) 16. A memory of 64 kilobytes holds  $64 \times 1024 = 65536$  separately addressed bytes, and 2 to the power 16 is exactly 65536, so 16 address lines are needed.
- (vi) It calls a constructor of the immediate parent class, passing on whatever arguments are given, so that the inherited part of the object is set up before the rest of the subclass constructor runs. It must stand first, and if it is left out the compiler inserts a call to the parent's constructor that takes no arguments.
- (vii) (a) public, static and final. Such a variable is a constant shared by every class that implements the interface, and the three modifiers are supplied by the compiler whether or not they are written out.
- (viii) A stack. Its elements leave in LIFO order, that is last in first out, the element added most recently being the first to be removed.
- (ix) (d) mesh. With every node joined to every other there is more than one path between any pair, so a single broken link only removes one of several routes.
- (x) (a) phishing. The message imitates a trusted sender in order to draw out credentials, which makes it a fraud worked on the reader rather than an attack on the machine.

*1 mark for each correct part. No mark is deducted for a wrong option, and a one-line answer that carries the correct idea gains the mark even if it is loosely worded.*

### Question 2 [10]

- (i) Octal. Group the bits in threes from the right, padding on the left: 010 110 110. The groups are 2, 6 and 6, so the octal value is  $(266)_8$ . Hexadecimal. Group the bits in fours from the right: 1011 0110. The group 1011 is 11, which is B, and 0110 is 6, so the hexadecimal value is  $(B6)_{16}$ . As a check, the binary value is  $128 + 32 + 16 + 4 + 2 = 182$ , and  $2 \times 64 + 6 \times 8 + 6 = 182$ , while  $11 \times 16 + 6 = 182$  as well.

*Octal 266 with the bits grouped in threes 1. Hexadecimal B6 with the bits grouped in fours 1.*

- (ii) Output, as produced by the compiled program: 6INFACE on the first line and ECAFNI6 on the second line. Working. The builder starts as INTERFACE, nine characters long. The call `delete(2, 5)` removes the characters at the indexes 2, 3 and 4, which are T, E and R, and stops before index 5, leaving INFACE, which is six characters long. The call `sb.length()` is then evaluated and gives 6, and `insert(0, 6)` places that 6 at the front, giving 6INFACE. The reverse call turns the builder round in place, giving ECAFNI6.

The first value begins with a digit because the length was taken after the deletion and then inserted into the builder itself, the int 6 being converted to the text 6 as it goes in.

*6INFACE with the deletion traced 1. ECAFN16 with the sentence explaining the leading digit 1.*

- (iii) Set the two numbers one above the other and add from the right. Rightmost column:  $1 + 1 = 10$ , so write 0 and carry 1. Second column:  $1 + 0$  plus the carry 1 = 10, so write 0 and carry 1. Third column:  $0 + 1$  plus the carry 1 = 10, so write 0 and carry 1. Leftmost column:  $1 + 1$  plus the carry 1 = 11, so write 1 and carry 1, and that final carry becomes a fifth bit. The sum is therefore  $(11000)_2$ . Check in decimal:  $(1011)_2 = 11$ ,  $(1101)_2 = 13$ , and  $11 + 13 = 24$ , while  $(11000)_2 = 16 + 8 = 24$ . The two agree.

*Addition carried out with the carry shown in each column, giving 11000 1. Decimal check  $11 + 13 = 24$  1.*

- (iv) Overloading means two or more methods in the same class carrying the same name but differing in the number or the types of their parameters, while overriding means a method in a subclass carrying the same name and the same parameter list as one in its parent and replacing it. Overloading is settled by the compiler from the arguments written at the call, which is why it is called early binding, while overriding is settled at run time from the class of the object the reference actually points at, which is late binding. A third acceptable difference is that overloaded methods may differ in return type, whereas an overriding method must keep a compatible return type and may not narrow the access of the method it replaces.

*One correct difference 1. A second correct difference 1.*

- (v) In a linear queue front and rear only ever move forward. Every removal advances front and leaves the place it vacated unused, so after six additions and two removals rear has reached the last place of the array while front stands at index 2. The test for a full queue looks only at rear, finds it at the end, and reports the queue as full even though the two places at the start are free. The waste is sometimes called the drift of a linear queue. The change that cures it is to make the queue circular, so that rear and front wrap round to index 0 once they pass the last place, the remainder operator being used on the capacity, and fullness being judged from a count of the elements present rather than from the position of rear.

*Reason that front leaves unused places behind while the full test looks only at rear 1. Circular queue named as the cure, with the count or the wrap round mentioned 1.*

### Question 3 [10]

- (a) The two theorems in symbols are  $(A + B)' = A'B'$  and  $(AB)' = A' + B'$ . In words, the complement of a sum is the product of the complements and the complement of a product is the sum of the complements.

For the proof of the first, draw a table of five columns, headed A, B,  $A + B$ ,  $(A + B)'$  and  $A'B'$ , with four rows. Row one,  $A = 0$  and  $B = 0$ :  $A + B = 0$ , so  $(A + B)' = 1$ , while  $A' = 1$  and  $B' = 1$ , so  $A'B' = 1$ . Row two,  $A = 0$  and  $B = 1$ :  $A + B = 1$ , so  $(A + B)' = 0$ , while  $A' = 1$  and  $B' = 0$ , so  $A'B' = 0$ . Row three,  $A = 1$  and  $B = 0$ :  $A + B = 1$ , so  $(A + B)' = 0$ , while  $A' = 0$  and  $B' = 1$ , so  $A'B' = 0$ . Row four,  $A = 1$  and  $B = 1$ :  $A + B = 1$ , so  $(A + B)' = 0$ , while  $A' = 0$  and  $B' = 0$ , so  $A'B' = 0$ . The fourth column and the fifth column agree in every row, which proves the theorem.

Note for the examiner: both theorems have been checked by machine over all four input combinations. Accept extra working columns for  $A'$  and  $B'$ .

*Both theorems stated in symbols 2. Truth table drawn with all four rows correct 1. The two columns seen to agree, with the conclusion drawn 1.*

- (b) Take the first two factors together.  $(A + B)(A + B') = A + BB'$  by the distributive law in the form  $(A + X)(A + Y) = A + XY$ , and  $BB' = 0$  by the complement law, so the pair reduces to  $A + 0$ , which is A by the law of identity. The expression is now  $A(A' + C)$ . Multiply out by the distributive law:  $A(A' + C) = AA' + AC$ . Now  $AA' = 0$  by the complement law, so the expression is  $0 + AC$ , which is AC by the law of identity. The simplest form is therefore  $F = A.C$ , which contains two literals.

Note for the examiner: this reduction has been checked by machine over all eight input combinations of A, B and C, and  $(A + B)(A + B')(A' + C)$  and  $A.C$  agree on every one. Accept a candidate who multiplies everything out into a sum of products first and then reduces, provided the laws are named.

*First two factors reduced to A 1. Distributive step giving  $AA' + AC$  and then AC 1. Laws named and the count of two literals 1.*

- (c) In eight bits,  $71 = 01000111$  and  $45 = 00101101$ . The 1's complement of 45 is 11010010, and adding 1 gives the 2's complement, 11010011. Now add:  $01000111 + 11010011 = 1\ 00011010$ , that is a carry of 1 out of the leftmost column with 00011010 in the eight bit result. The carry is discarded, and its presence shows the answer is positive. The result 00011010 is  $16 + 8 + 2 = 26$ , so 71 minus 45 = 26.

The method signals a negative answer by producing no carry out of the leftmost column, which leaves the result with its leading bit at 1. For 25 minus 58:  $25 = 00011001$ ,  $58 = 00111010$ , the 2's complement of 58 is  $11000110$ , and  $00011001 + 11000110 = 11011111$ , with no carry out. The leading bit is 1, so the answer is negative, and its magnitude is found by taking the 2's complement of the result, which is  $00100001$ , that is  $32 + 1 = 33$ . So 25 minus 58 = minus 33.

Note for the examiner: both subtractions have been checked numerically.

*Eight bit forms with the 2's complement of 45, and the addition 1. Carry discarded and the answer 26 1. No carry out recognised as the sign of a negative answer, with the magnitude 33 obtained by re-complementing 1.*

#### Question 4 [10]

- (a) List the combinations as binary values of ABCD. The function is 1 at 0000, 0001, 0100, 0101, 0110, 0111, 1001 and 1101. Draw a four variable Karnaugh map as a four by four grid, the columns headed by CD in the Gray code order 00, 01, 11, 10 and the rows headed by AB in the same order 00, 01, 11, 10, and place a 1 in each of those eight squares.

Three groups cover every 1. First group, a block of four made of the squares 0000, 0001, 0100 and 0101. In all four A is 0 and C is 0, while B and D each take both values, so the term is  $A'C'$ . Second group, a block of four made of the squares 0100, 0101, 0110 and 0111, which is the whole  $AB = 01$  row. In all four A is 0 and B is 1, so the term is  $A'B$ . Third group, a block of four made of the squares 0001, 0101, 1101 and 1001, which is the whole  $CD = 01$  column. In all four C is 0 and D is 1, so the term is  $C'D$ .

The simplified sum of products form is therefore  $F = A'C' + A'B + C'D$ .

Note for the examiner: this reduction has been checked by machine over all sixteen input combinations and agrees with the given function on every one. Several squares are covered more than once, which is allowed and is what lets each group reach its full size of four. Accept the three terms written in any order.

*Map drawn with Gray code ordering on both edges and the eight squares marked 2. Group of four giving  $A'C'$  1. Group of four giving  $A'B$  1. Group of four giving  $C'D$  1.*

- (b) The zeros lie at 0010, 0011, 0110, 0111, 1000, 1001, 1100 and 1101. Draw the same four by four map and place a 0 in each of those eight squares. Each group of zeros yields a sum term in which a variable appears complemented if it is 1 throughout the group and uncomplemented if it is 0 throughout.

Two groups of four cover all eight zeros. First group, the squares 0010, 0011, 0110 and 0111. In all four A is 0 and C is 1, while B and D each take both values, so the sum term is  $A + C'$ . Second group, the squares 1000, 1001, 1100 and 1101. In all four A is 1 and C is 0, so the sum term is  $A' + C$ .

The simplified product of sums form is therefore  $G = (A + C')(A' + C)$ . Multiplied out this is  $AC + A'C'$ , which is 1 exactly when A and C are the same, so the function is the exclusive NOR of A and C. The single two input gate that performs it is the XNOR gate.

Note for the examiner: this reduction has been checked by machine over all sixteen combinations. Note that B and D drop out altogether, the function depending only on A and C. Accept the two factors in either order, and accept the gate named as an exclusive NOR or as an equivalence gate.

*Map of zeros drawn and the eight squares marked 2. Group giving  $A + C'$  1. Group giving  $A' + C$  1. XNOR named as the gate 1.*

#### Question 5 [10]

- (a) A full adder is the combinational circuit that adds three single binary digits and produces a sum bit and a carry out bit, so that it can serve any column of a multi bit addition. Its three inputs are the two digits of the column, A and B, and the carry in from the column to its right,  $C_{in}$ .

The truth table should be drawn with five columns, A, B,  $C_{in}$ , Sum and  $C_{out}$ , and eight rows. With all three inputs 0 the sum is 0 and the carry out is 0. With  $A = 0$ ,  $B = 0$  and  $C_{in} = 1$  the sum is 1 and the carry out is 0. With  $A = 0$ ,  $B = 1$  and  $C_{in} = 0$  the sum is 1 and the carry out is 0. With  $A = 0$ ,  $B = 1$  and  $C_{in} = 1$  the total is 2, so the sum is 0 and the carry out is 1. With  $A = 1$ ,  $B = 0$  and  $C_{in} = 0$  the sum is 1 and the carry out is 0. With  $A = 1$ ,  $B = 0$  and  $C_{in} = 1$  the total is 2, so the sum is 0 and the carry out is 1. With  $A = 1$ ,  $B = 1$  and  $C_{in} = 0$  the total is 2, so the sum is 0 and the carry out is 1. With all three inputs 1 the total is 3, so the sum is 1 and the carry out is 1. The rule throughout is that the sum is the total counted in binary and the carry out is the second bit of

that total.

The sum output is  $\text{Sum} = A \text{ XOR } B \text{ XOR } C_{in}$ , which is 1 whenever an odd number of the three inputs is 1.

Note for the examiner: the table and the expression have been checked by machine against the arithmetic over all eight input combinations.

*Full adder defined with the three inputs named 1. Truth table drawn with the sum column correct in all eight rows 1. Carry out column correct in all eight rows 1. Sum expression as the exclusive OR of the three inputs 1.*

- (b) Volatility. Primary memory, in the form of the main random access memory, is volatile and loses its contents the moment the power goes, while secondary memory such as a hard disc or a solid state drive holds its contents without power and is where everything is kept between one session and the next. The way the processor reaches the contents. The processor addresses primary memory directly, fetching a word over the address and data buses in a few of its own cycles, while it cannot reach secondary memory directly at all. The contents of a file must first be copied into primary memory by the operating system before any instruction can work on them. Cost for each byte. Primary memory costs considerably more for each byte stored, which is why a machine carries a few gigabytes of it alongside hundreds of gigabytes or more of secondary storage.

Cache memory is fitted because primary memory, fast as it is, is still slower than the processor, so the processor would spend much of its time waiting. The cache is a small quantity of much faster memory sitting between the two, and it holds copies of the instructions and data most recently used. Because programs tend to return to the same instructions and the same data again and again, most requests are met from the cache at full speed, and only those that miss have to go out to the slower main memory.

*Volatility distinguished 1. Direct addressing against access through a copy into main memory, with the relative cost for each byte 1. Cache explained as closing the speed gap, with locality of reference 1.*

- (c) The mask 255.255.255.192 is 11111111.11111111.11111111.11000000 in binary, so 26 bits are given to the network part and 6 bits are left for the host part. Six host bits give  $2^6$ , that is 64 addresses in the block, running from 192.168.10.0 to 192.168.10.63. Two of those cannot be given to a machine, the first, 192.168.10.0, being the address of the network itself and the last, 192.168.10.63, being the broadcast address, so 62 addresses are available for machines, namely 192.168.10.1 to 192.168.10.62. The broadcast address of the block is therefore 192.168.10.63. The protocol that hands out addresses automatically as machines join is DHCP, the dynamic host configuration protocol.

Note for the examiner: these figures have been checked by machine. Accept the two reserved addresses named in either order.

*Six host bits with 64 total addresses 1. 62 usable addresses, with the two reserved ones identified 1. Broadcast address 192.168.10.63 and DHCP named 1.*

## Question 6 [10]

- (a) The completed class is as follows.

```
public class Hostel
{
 private String student;
 private int units;
 private double bill;
 public Hostel(String s, int u)
 {
 student = s;
 units = u;
 bill = 0.0;
 }
 public void computeBill()
 {
 if (units <= 100)
```

```

 bill = units * 4.0;
 else if (units <= 300)
 bill = 400.0 + (units - 100) * 6.5;
 else
 bill = 1700.0 + (units - 300) * 8.0;
 if (bill > 1500.0)
 bill = bill + bill * 5 / 100;
}
public void display()
{
 System.out.println("Student : " + student);
 System.out.println("Units : " + units);
 System.out.println("Bill : " + bill);
}
public static void main(String args[])
{
 Hostel p = new Hostel("Riya Sharma", 80);
 p.computeBill();
 p.display();
 Hostel q = new Hostel("Imran Qureshi", 250);
 q.computeBill();
 q.display();
 Hostel r = new Hostel("Tara Nair", 400);
 r.computeBill();
 r.display();
}
}

```

Working for the three sets of data. For 80 units the first rule applies, giving  $80 \times 4.0 = 320.0$ , which does not exceed 1500.0, so no surcharge is added. For 250 units the second rule applies, giving  $400.0 + 150 \times 6.5 = 400.0 + 975.0 = 1375.0$ , which again does not exceed 1500.0, so no surcharge is added. For 400 units the third rule applies, giving  $1700.0 + 100 \times 8.0 = 2500.0$ , which does exceed 1500.0, so the surcharge of 5 per cent, that is 125.0, is added, giving 2625.0.

Output, as produced by the compiled program, is these nine lines in order: Student : Riya Sharma, then Units : 80, then Bill : 320.0, then Student : Imran Qureshi, then Units : 250, then Bill : 1375.0, then Student : Tara Nair, then Units : 400, then Bill : 2625.0.

Note for the examiner: the surcharge test must be made after the charge has been worked out and must stand outside the three branches, not inside the last one, since the wording ties it to the amount and not to the band. On these three sets of data only the third crosses the threshold, but a candidate who writes the test inside the else branch loses the mark even though the printed figures happen to agree. The bill must carry its decimal point, since the field is of type double. Accept  $\text{bill} * 0.05$  in place of  $\text{bill} * 5 / 100$ , since both give 125.0 in double arithmetic.

*Class header with the three private data members 1. Parameterised constructor setting all three fields 2. First and second bands of the rule correct 2. Third band correct 1. Surcharge of 5 per cent applied after the bands and only above 1500.0 2. display printing the three lines in the right order 1. main creating all three objects, with the nine output lines correct 1.*

### Question 7 [10]

- (a) Dry run. Before the loop  $p$  is 1 and  $q$  is 9. The test 1 less than 9 passes. First pass:  $p = 1 \times 2 = 2$ , then  $q = 9 - 2/2 = 9 - 1 = 8$ , and (2,8) is printed. The test 2 less than 8 passes. Second pass:  $p = 2 \times 2 = 4$ , then  $q = 8 - 4/2 = 8 - 2 = 6$ , and (4,6) is printed. The test 4 less than 6 passes. Third pass:  $p = 4 \times 2 = 8$ , then  $q = 6 - 8/2 = 6 - 4 = 2$ , and (8,2) is printed. The test 8 less than 2 now fails, so the loop ends.

Output, as produced by the compiled program: the first line reads (2,8) (4,6) (8,2) followed by a trailing space, since print adds a space after every pair, and the second line reads 10.

The loop body is executed three times. Note that  $q$  is worked out using the new value of  $p$ , because  $p$  is doubled first, and that  $p / 2$  is integer division, which here is exact at every pass.

*The three passes traced with both values at each 1. First line (2,8) (4,6) (8,2) 1. Second line 10 with three executions stated 1.*

- (b) Output, as produced by the compiled program, is these six lines in order: Shape built with side 3, then Cube built, then 54, then Shape built with side 2, then Cube built, then 8.

Working. The statement `new Cube(3)` runs the Cube constructor, whose first act is `super(s)`, so the Shape constructor runs first and prints Shape built with side 3, after which Cube built is printed. The call `ob.measure()` then gives  $6 \times 3 \times 3 = 54$ . Creating `new Cube(2)` prints the same two lines with the side 2. The call `cb.baseMeasure()` reaches the parent version through `super.measure()`, which gives  $4 \times 2 = 8$ .

The first call to measure gives 54 rather than 12 because measure is overridden in Cube, and an overridden method is chosen at run time from the class of the object the reference actually points at, not from the declared type of the reference. The variable `ob` is declared as a Shape but holds a Cube, so the Cube version runs. This is run time polymorphism.

*All four constructor lines in the right order, parent before child 1. 54 1. 8 from super.measure() 1. Explanation by run time polymorphism, the object's class deciding and not the reference type 1.*

- (c) The method `g` adds the last digit of  $n$  to the result of the same method called on  $n$  with that digit removed, so it sums the digits of the number. For `g(4729)`:  $9 + g(472) = 9 + 2 + g(47) = 9 + 2 + 7 + g(4) = 9 + 2 + 7 + 4 + g(0)$ , and `g(0)` returns 0, so the value is 22. The method `h` divides  $n$  by 2 and places the remainder behind whatever the recursive call returns, so it builds the binary representation of  $n$  as a string. For `h(45)`: `h(22) + 1`, and `h(22) = h(11) + 0`, and `h(11) = h(5) + 1`, and `h(5) = h(2) + 1`, and `h(2) = h(1) + 0`, and `h(1) = h(0) + 1 = "" + 1 = "1"`. Working back out: `h(2) = "10"`, `h(5) = "101"`, `h(11) = "1011"`, `h(22) = "10110"`, `h(45) = "101101"`.

Output, as produced by the compiled program: 22 on the first line and 101101 on the second line.

So `g` computes the sum of the digits of its argument, and `h` computes the binary representation of its argument as a string. As a check, 101101 in binary is  $32 + 8 + 4 + 1 = 45$ . Note that the remainder in `h` is joined to the string rather than added, because the left operand of the plus sign is already a String.

*22 with the chain of digit sums 1. 101101 with the recursion unwound 1. What each method computes, stated for both 1.*

## Question 8 [10]

- (a) The program is as follows.

```
import java.util.Scanner;

public class NameSort
{
 public static void main(String args[])
 {
 Scanner in = new Scanner(System.in);
 String n[] = new String[8];
 System.out.println("Enter eight names");
 for (int i = 0; i < 8; i++)
 n[i] = in.next();
 for (int i = 0; i < 7; i++)
 {
```

```

 for (int j = 0; j < 7 - i; j++)
 {
 if (n[j].compareTo(n[j + 1]) > 0)
 {
 String t = n[j];
 n[j] = n[j + 1];
 n[j + 1] = t;
 }
 }
 }
}

System.out.println("Names in alphabetical order");
for (int i = 0; i < 8; i++)
 System.out.println(n[i]);
System.out.println("Enter the name to be searched");
String key = in.next();
int pos = -1;
for (int i = 0; i < 8; i++)
{
 if (n[i].equals(key))
 {
 pos = i;
 break;
 }
}
if (pos == -1)
 System.out.println(key + " is not in the list");
else
 System.out.println(key + " found at position " + (pos + 1));
}
}

```

Output, as produced by the compiled program on the given data. The line Enter eight names appears first as the prompt. Then Names in alphabetical order, followed by the eight names one to a line in this order: Anita, Bhavna, Deepa, Kavya, Nitin, Rahul, Suresh, Vikram. Then the prompt Enter the name to be searched, and finally the line Nitin found at position 5.

Note for the examiner: compareTo returns a negative number, zero or a positive number according as the first string sorts before, with or after the second, so a result greater than zero is the signal to swap. The relational operators would not work here, since they would compare references rather than text. The position is reported as pos + 1 because the array is indexed from zero but the question asks for a count from one. Had the name searched for been one not in the list, such as Mohan, the last line would read Mohan is not in the list, which has also been checked. Accept an inner loop written as j less than 7 without the minus i, since the result is the same although more comparisons are made.

*Array of eight String declared and the input loop 1. Nested loops of a bubble sort with the inner range shrinking 1. Comparison made with compareTo and a result greater than zero taken as the signal to swap 1. Three statement swap through a temporary String 1. Sorted names printed one to a line in the correct order 1. Search loop using equals, with the position reported from one and the not found case handled 1.*

**(b)** The program is as follows.

```
import java.io.*;
```

```

public class FileWord
{
 public static void main(String args[])
 {
 try
 {
 PrintWriter pw = new PrintWriter(new FileWriter("notes.txt"));
 pw.println("Data travels as packets");
 pw.println("Every packet carries an address");
 pw.close();
 BufferedReader br = new BufferedReader(new FileReader("notes.txt"));
 String line;
 int lines = 0, words = 0, longOnes = 0;
 while ((line = br.readLine()) != null)
 {
 lines = lines + 1;
 String part[] = line.split(" ");
 words = words + part.length;
 for (int i = 0; i < part.length; i++)
 {
 if (part[i].length() > 5)
 longOnes = longOnes + 1;
 }
 }
 br.close();
 System.out.println("Lines = " + lines);
 System.out.println("Words = " + words);
 System.out.println("Words longer than five letters = " + longOnes);
 }
 catch (IOException e)
 {
 System.out.println("File error : " + e.getMessage());
 }
 }
}

```

Working. The file holds two lines. The first, Data travels as packets, splits into four words, of which travels and packets are longer than five letters, giving two. The second, Every packet carries an address, splits into five words, of which packet, carries and address are longer than five letters, giving three. So the totals are two lines, nine words and five long words.

Output, as produced by the compiled program: Lines = 2, then Words = 9, then Words longer than five letters = 5, each on a line of its own.

Note for the examiner: the writer must be closed before the reader is opened, or the buffered text may not yet have reached the file and the counts come out as zero. The test is length greater than 5, so a word of exactly five letters, such as Every, is not counted. Accept a StringTokenizer in place of split, and accept the catch written for Exception rather than IOException.

*File written with two lines and the writer closed 1. File reopened and read line by line until readLine returns null 1. Lines, words and long words counted correctly 1. Whole of the file handling inside a try with a catch, and the three output lines correct 1.*

**Question 9 [10]**

(a) The completed class is as follows.

```
public class Ring
{
 private int q[];
 private int front, rear, count, cap;
 public Ring(int n)
 {
 cap = n;
 q = new int[cap];
 front = 0;
 rear = -1;
 count = 0;
 }
 public void addRear(int v)
 {
 if (count == cap)
 System.out.println("Queue is full, cannot add " + v);
 else
 {
 rear = (rear + 1) % cap;
 q[rear] = v;
 count = count + 1;
 }
 }
 public int removeFront()
 {
 if (count == 0)
 {
 System.out.println("Queue is empty");
 return -1;
 }
 int v = q[front];
 front = (front + 1) % cap;
 count = count - 1;
 return v;
 }
 public static void main(String args[])
 {
 Ring r = new Ring(4);
 r.addRear(7);
 r.addRear(14);
 }
}
```

```

 r.addRear(21);
 System.out.println(r.removeFront());
 System.out.println(r.removeFront());
 r.addRear(28);
 r.addRear(35);
 r.addRear(42);
 r.addRear(49);
 System.out.println(r.removeFront());
 System.out.println(r.removeFront());
 System.out.println(r.removeFront());
 System.out.println(r.removeFront());
 System.out.println(r.removeFront());
 }
}

```

Working. The capacity is four. Adding 7, 14 and 21 fills the places 0, 1 and 2, so rear is 2, front is 0 and count is 3. The first two removals return 7 and 14, leaving front at 2 and count at 1. Adding 28 puts it at place 3, and adding 35 sends rear round to place 0, since  $(3 + 1)$  remainder 4 is 0, so 35 occupies the place 7 vacated. Adding 42 puts it at place 1 and brings count to 4, which is the capacity. The add of 49 therefore finds the queue full and reports it without storing anything. The five removals that follow return 21, 28, 35 and 42 in that order, after which count is 0, so the fifth removal reports the queue as empty and returns minus 1.

Output, as produced by the compiled program, is these nine lines in order: 7, then 14, then Queue is full, cannot add 49, then 21, then 28, then 35, then 42, then Queue is empty, then -1.

The remainder operator makes front and rear wrap round to index 0 once they pass the last place of the array, so the places freed by earlier removals are used again instead of being wasted.

Note for the examiner: the full line appears third because the adds all run before the later removals are printed.

The count member is what makes a full queue distinguishable from an empty one, since in both cases front and rear can stand in the same relation to each other. Accept a solution that leaves one place always empty in place of keeping a count, provided the full and empty tests are consistent and the printed values agree.

*Five private data members with the constructor sizing the array and setting front, rear and count 2. addRear testing count against the capacity and advancing rear with the remainder operator 2. removeFront testing for an empty queue, returning minus 1 with a message, and advancing front in the same way 2. All nine output lines correct and in order, with the sentence on the remainder operator 1.*

- (b) A linked list gives no way to reach a node except by starting at head and following the link field of each node in turn, because the nodes are scattered through memory and nothing records where the nth one lies. Counting them therefore means walking the whole chain, one node at a time, so the work grows in proportion to the length of the list. An array, by contrast, occupies one continuous block of memory in which every element takes the same number of bytes, so the address of the element at a given index can be worked out in one arithmetic step from the address of the first element, the index and the element size. That calculation costs the same whatever the index, which is why an array is said to allow random access.

In return, a linked list can grow and shrink as it is used, needing no decision about its size in advance and no continuous block of free memory, and an insertion or a deletion in the middle of it costs only the resetting of a link or two, whereas the same operation in an array means shifting every element beyond the point of change.

*Reason that a linked list must be walked from head, with the cost growing with its length 1. Address arithmetic in an array giving the same cost at any index 1. One advantage of the linked list in return 1.*

### Question 10 [10]

- (a) Intellectual property in software means the rights that belong to whoever created an original piece of software, covering the code they wrote, the design they worked out and the name under which it goes out, and giving them the say over how it is copied, changed, sold and credited. The rights exist in the work itself, not in

the disc or the download that carries it.

Copyright protects the expression, that is the source code and the object code as a piece of original writing, together with the manuals, the screen text and the artwork. It arises automatically the moment the code is written and governs copying, adaptation and distribution, but it does not stop anybody writing their own program that does the same job. A patent protects an invention, that is a genuinely new and non obvious technical method the software embodies, and where it is granted it stops others using that method even in code they wrote themselves. It must be applied for and runs for a limited term. A trademark protects the name, the logo and the look under which the software is sold, so that a customer can tell whose product is in front of them, and it stops a rival passing off their own program under a confusingly similar name.

Free and open source software is software released with its source code and with a licence that lets anyone study it, run it for any purpose, change it and pass on the original or the changed version. The word free refers to that liberty rather than to the price, and many such licences require that anything built on the work be released on the same terms, so that the freedom travels with the code. It is not the same as software that is merely given away without charge, which may still keep its source closed.

Copying a few paragraphs from another person's blog into a project report without naming the source is wrong whether or not money changes hands, for two separate reasons. It infringes the writer's copyright, which subsists in the words regardless of what the copier gains. It is also plagiarism, which is the offence of presenting someone else's work as your own, and in a report submitted for assessment that is a misrepresentation to the examiner about whose thinking is being marked. Both are cured by quoting the passage and naming the source, which costs nothing.

*Intellectual property in software explained 1. Copyright and what it protects 1. Patent and trademark, each with what it protects 1. Free and open source software explained, with the point that free refers to liberty 1. Copying without attribution identified as both an infringement of copyright and plagiarism 1.*

- (b) Big data means collections of data so large, so fast arriving or so mixed in kind that the ordinary tools, a single database on a single machine, can no longer store or process them, so that the work must be spread over many machines. The usual three properties are volume, that is the sheer quantity, measured in terabytes and upwards, velocity, that is the rate at which new data arrives and the speed at which it must be handled, often as a continuous stream rather than a batch, and variety, that is the mixture of kinds, structured records sitting alongside text, images, video and sensor readings that fit no fixed set of columns. Veracity, meaning how far the data can be trusted, and value, meaning whether anything useful can be got out of it, are often added as a fourth and fifth.

Holding a very large quantity of personal data is a risk to the holder as well as to the people described, because the collection itself becomes a target worth attacking and the holder carries the legal duty and the loss of trust if it leaks, so that data gathered on the chance it may prove useful is a liability sitting on the books until the day it is either used or deleted.

*Big data defined by the inadequacy of ordinary tools 1. Three properties named and explained 1. Risk to the holder as well as to the people described 1.*

- (c) A blockchain is a record kept as a chain of blocks, each block holding a batch of entries together with a fingerprint of the block before it, the whole chain being held in full on many computers at once rather than by any single keeper, and a new block being added only when the participants agree by an agreed procedure that it is valid.

The property that makes an entry hard to alter quietly is that each block carries the fingerprint of its predecessor. Changing anything in an old block changes that block's own fingerprint, which breaks the link held by the block after it, and every block from there to the end of the chain would have to be recomputed to hide the change. Since every participant holds a copy and the chain grows continuously, that recomputation would have to beat the honest copies held everywhere else, which is why the record is called tamper evident.

*Blockchain described as a chain of blocks held by many participants, each block linked to the one before 1. Alteration broken by the chain of fingerprints across the copies 1.*