2025 ASSESSMENT REPORT
ITC315118 COMPUTER SCIENCE

Section A – Criterion 1

Question 1

a. Almost all students achieved full marks.

b. Most students answered this question correctly. Part marks were given for the answer “10”.

c.

i. Very well done. Some students lost marks for not stating the line numbers to be edited.

ii. Around half of all students gained full marks for this question. A common mistake was to keep rewarding the bonus payout **every time** the user inserted a bottle or can after $1.

Question 2

a. Around two-thirds of students achieved full marks for this question. Several students put a “9” in the last row of the table, which would not have occurred with the algorithm.

b. To achieve full marks for this part, students needed to include “List3” in line 2, as well as repeat the four lines of code to successfully search List3. Less than half of students achieved full marks here.

c. Almost half of students achieved full marks for this question. Many students took the opportunity to describe the method for altering the algorithm, rather than providing complete code, which was pleasing to see. Solutions included:

- adding “else if” statements and increasing “pos” within each “if” clause
- simply increasing “pos” within the “if” statement, then removing it at the end
- introducing a boolean variable that keeps track of whether the next value has been found.

Question 3

90% of students attempted this question, which was a significant increase from previous years. Marks were evenly spread across the whole range. Some excellent answers were presented.

There were two different models employed by students when implementing this algorithm. The difference was whether the components of a meal were all turned off or all turned on when a meal was turned on. Which model was used affected how the Burger, SmallMeal and LargeMeal button algorithms were written. The solutions below turned off all the components of a meal, which was the more popular model employed by students.

Note that line numbers were not required in solutions.

Marks were allocated for each of the required sections for the following concepts:

When the burger button was turned on:

- Students needed to consider if including a burger now made a small or large meal. This included perhaps upgrading a small and large item to a large meal. This all could be done here or in the calculate section.

Small meal:

- When it is turned off
  - Whether all the components of the meal were turned off or on.
- When it is turned on
  - Whether all the components of the meal were turned off or on
  - Turning off the large meal button
  - Turning off all the large meal components.

Large meal:

- When it is turned off
  - Was it a small and large item that made the meal in the first place? This was an issue when the model used was to leave all the components on.
- When it is turned on
  - Whether all the components of the meal were turned off or on
  - Turning off the small meal button
  - Turning off all the small meal components.

Calculate button:

- Students needed to ensure that every item was appropriately charged, especially not charging for a meal component and the meal price itself.

Section B – Criterion 2

Question 4

a. Almost all students got full marks.

b.

i. About one-third of students had difficulty with integer division and the remainder (%) function.

ii. Many students did not use the substring() method correctly.

iii. About one-third of students had difficulty with the increment and decrement operators (b++, c--).

iv. Almost all students got full marks.

Question 5

a.

i. About two-thirds of students completed the trace table successfully.

ii. About two-thirds of students completed the trace table successfully.

iii. Inclusion of the Math.sqrt() method into the loop in Question 5a (iii) was correctly applied by one-third of students, one-third received partial marks and one-third did not attempt this question.

iv. A little over one-third of students were able to change the method to make it more efficient, one-third did not attempt this question.

b. About one-third of students did not attempt this question. About half of those that did, scored full marks, with the rest awarded partial marks.

Question 6

It was pleasing that around half of all students attempted parts of Question 6 which was more than in past years.

a. About half completed the trace table successfully.

b. Less than half the students attempted to explain when the **findSeats()** method would return TRUE. Most of those that did were awarded full marks.

c. Most students could not identify that variable **people** would need to be passed as a parameter if it were no longer declared globally.

d. About one-third of students added the correct code.

e. About one-third of students completed the trace table correctly.

Section C – Criterion 2

Question 7

a. Almost all students gained full marks.

b. Almost all students gained full marks.

c.

i. About two-thirds of students gained full marks. Instead of declaring the Rectangle object as required, a few students provided code to draw a rectangle.

ii. More than half of all students gained full marks for this question. The most common errors were related to the syntax of declaring an object.

iii. Half of all students gained full marks for this question. For full marks it was only necessary to demonstrate understanding of how to pass

Question 8

a. This was generally answered well, with about two-thirds of students gaining full marks. Other than general confusion about how to declare objects, the most common mistake was omitting quote marks ("") around the string literals.

b. About two-thirds of students gained full marks.

c. About half of students gained full marks. A common interpretation of this and the following question was to determine the actual distance rather than write a Java expression that would return the distance. Unfortunately, very few responses determined the distances correctly. Partial marks were awarded for working out the demonstrated calculations required.

Responses that included correct Java expressions were given full marks regardless of subsequent algebraic and numerical answers.

d. About a quarter of students gained full marks.

e. Very few students obtained full marks. The most common mistake was omitting setter methods for the latitude and longitude.

Question 9

a. More than two-thirds of students attempted this question. A common error was to calculate the strike rate using integer division.

   Marks were awarded for the following:

   - correct class structure, including declaration of variables and their initialisation via the required constructor (2 marks)
   - correct method structure, including use of parameters (2 marks)
   - dealing with the strike rate, including use of floating-point division to correctly determine the value (2 marks).

b. Less than half of all students attempted this question. For full marks, students were expected to write code that would use methods efficiently and robustly. A common mistake was to compare strings using the "==" operator, which compares whether the variables reference the same *object*, but not whether the strings have the same *value*.

   Marks were awarded for the following:

   - loop through the array and output the results (2 marks)
   - filter and count the matching ground (2 marks)
   - display "yes" or "no" rather than "true" or false" (2 marks).

Section D – Criterion 4

Question 10

a.

   i. Answered correctly by about 80% of students.
   ii. Answered correctly by almost all students (93%).
   iii. Answered correctly by almost all students (99%).
   iv. Answered correctly by almost all students (94%).

b. Answered correctly by most students (90%).

c.

   i. Answered correctly by most students (89%).
   ii. About two-thirds of students (65%) answered this correctly. Some students lost marks for not fully demonstrating the non-equivalence or incorrectly completing parts of the truth table.

Question 11

a. Answered correctly by about three-quarters of students, with 76% receiving full marks. About 23% of students could not explain why the memory address contained data rather than an instruction to be executed. These students either left the question blank or provided insufficient explanation about the distinction between data storage and executable instructions.

b.

i. About two-thirds of students answered this satisfactorily, with about 35% achieving full marks. About 58% of students scored 3-4 out of 4, indicating that most could complete the basic K-map grouping but struggled with optimisation or with deriving the final simplified Boolean expression.

Students lost marks for:

- incomplete K-map grouping (not identifying the largest possible groups)
- creating invalid groups (non-rectangular or not powers of 2)
- deriving incorrect Boolean equations from their groupings
- stopping before full simplification.

ii. Answered correctly by about three-quarters of students, with about 53% receiving full marks. Most students who correctly applied the logic laws received full marks. Those who received partial marks typically stopped halfway with their simplifying and could have gone further or made errors in applying De Morgan's Laws or distribution laws. Some students lost marks for using brackets but not adjusting the AND/OR statement when placing the NOT outside of the brackets.

c. Less than half of students got this correct, with only about 10% achieving full marks.

This was the most challenging part of Question 11. Many students showed some working but could not reach the final simplified form. Students lost marks for:

- not fully simplifying the logic expressions
- stopping halfway through the simplification process
- using brackets but not adjusting the AND/OR statement when placing the NOT outside of the brackets
- incorrect application of De Morgan's Laws
- errors in applying the distribution or absorption laws.

Question 12

This question proved very challenging for most students. About 4% of students achieved full marks across all of Question 12.

a.

i. Of those who attempted, about 71% were able to trace the bitwise AND operation correctly.

ii. Less than half of the students got this correct. Of those who attempted, most were able to trace correctly but then lost marks for the final execution steps or the calculation of the final result.

iii. Only about 26% of students got this correct. Most students could not identify that the program counts the number of 1-bits in the binary representation of the number. Common incorrect responses included "divides by 2", "figures out the remainder", or "counts down until zero". Those who scored partial marks typically described only part of the program's function (such as "shifts right" or "counts loops") without identifying the overall purpose of counting set bits.

b. Half the students scored zero marks on this question. Only about 16% achieved 4 or more marks out of 5.

Common errors included:

- infinite loops from using R[0]==0 as an exit condition (R[0] always contains zero)
- invalid output format (using 9X02, 9X01, or 9X03 instead of 9XFF)
- off-by-one errors in loop counters (wrong number of iterations)
- failure to update both Fibonacci values correctly each iteration
- uninitialised registers being used in calculations
- incomplete instructions (only 1-2 hex digits instead of 4)
- attempting to use invalid hex digits beyond F (such as G, H, or Z)
- attempting to write to R[0] which is hardwired to zero
- missing output instruction entirely or output instruction unreachable due to infinite loop.

Section E – Criterion 5

Question 13

a. Almost all students gained full marks. The only notable exception was Question 13a iii. Some students were not too sure how to carry the 1.

b.

i. Almost all students gained full marks.
ii. Almost all students gained full marks.
iii. Almost all students gained full marks.
iv. About two-thirds of students gained full marks.
v. About three-quarters of students gained full marks.

Question 14

a. About two-thirds of students gained full marks. The most common incorrect answer was 6.

b.

i. More than two-thirds of students answered this question correctly. Common mistakes included not identifying a negative sign, incorrectly calculating the exponent, and/or incorrectly determining the mantissa (e.g. 0.73).
ii. The preferred solution for this question was to represent the floating-point number in normalised form (where the mantissa is adjusted so the first non-zero digit appears immediately after the radix point, and the exponent is altered accordingly). Many students did not normalise their floating-point number; however, part marks were awarded if their unnormalised representation still produced a correct result.

iii. This question was not answered well by most students, with only a quarter gaining full marks. There seemed to be some confusion between the mantissa and exponent, with some students interchanging these terms.

c. Answered correctly by about two-thirds of students.

Question 15

a. This question was answered poorly by more than half of students. The most noticeable aspect was students writing down calculations without any explanation. For example, when a response includes "10 × 6" without any other context, it's not clear what calculation this is for. Responses consisting of multiple lines of unexplained calculations resulted in part marks at best. Students who provided a solution with explanations of every calculation received at least 4 marks.
For questions that involve multiple calculations, students should be reminded to label each step.

b.
   i. This question was not answered well by 80% of students.
   ii. This question was not answered well by 90% of students.
   iii. This question was attempted by 70% of students. Given that the definition of lossless compression was provided in the Information Booklet, it was pleasing to see a diverse range of responses demonstrating a solid understanding of the concept.

Sample Solutions

Section A

Question 1

a.
   i. 1
   ii. 4
   iii. BA
   iv. 11

b. 11

c.
   i. Edit line 6 to say:

```
6 Payout = Payout + $0.10
```

   ii. Add after line 7:

```
7a    if Payout = $1.00
7b       Payout = Payout + $0.50
```

Question 2

a.

ListFinal	
Position	Name
1	Tony
2	Harry
3	Megan
4	Frank
5	Tina
6	Josh
7	John
8	Chloe

b. Change Line 2:

```
2  While List1 or List2 or List3 are not empty
```

After line 10 add:

```
10a  search List3
10b  if pos is found in List 3
10c    remove pos and the corresponding name from List 3
10d    add position and corresponding name to next row of ListFinal
```

c. "Else if" statements are added after each "if clause" and "pos" is incremented within each clause.

After line 6 add:

```
6a  increase pos by 1
6b  else if
```

Before line 10a (from above) add:

```
10i  increase pos by 1
10ii else if
```

After line 10d add (within the "if" clause):

```
10e  increase pos by 1
```

Remove line 11.

Question 3

```
Initially
 cost = $0.00
 turn off all buttons

When the Burger button is pressed
 if the Burger button is now off
  turn off Large Meal button
  turn off Small Meal button
 else
  if Small Chips button is on and Small Drink button is on
   turn on Small Meal button
   turn off Large Chips button
   turn off Large Drink button
  else if (Large Chips button is on and Small Drink button is on) or
      (Large Chips button is on and Large Drink button is on) or
      (Small Chips button is on and Large Drink button is on)
   turn on Large Meal button
   turn off Small Chips button
   turn off Small Drink button
   turn off Small Meal button
  end if
 end if

When the Small Meal button is pressed
 if the Small Meal button is on
  turn off all other buttons
  turn on Burger button
  turn on Small Drink button
  turn on Small Chips button
 end if

When the Large Meal button is pressed
 if the Large Meal button is on
  turn off all other buttons
  turn on Burger button
  turn on Large Drink button
  turn on Large Chips button
 end if

When the Calculate button is pressed
 if the Small Meal button is on
  cost = $14.50
 else if the Large Meal button is on
  cost = $15.50
 else
  cost = 0
  if the Burger button is on
```

```
  cost = $10.00
 end if
 if the Small Chips button is on
  cost += $3.00
 else if the Large Chips button is on
  cost += cost + $4.00
 end if
 if the Small Drink button is on
  cost += cost + $2.50
 else if the Large Drink button is on
  cost += cost + $3.50
 end if
end if
display cost
```

Section B

Question 4

a.

i. `Char`

ii. `13`

iii. `'B'`

iv. `3.0`

b.

i. `b = 2 + 1 = 3`. "/" is integer division and "%" computes the remainder.

ii. = "ab" + 'C' + "de" = "abCde"

`The first two characters of s, with the character 'C', with the last two characters of s are concatenated to give the answer.`

iii. b = 4, c = 5, then the value of c is added to b

b = 9

iv. j = 25

j
3
5
15
25

Question 5

a.

i.

n	j	n % j	prime	isPrime(7)
7	2	1	true	
	3	1		
	4	3		
	5	2		
	6	1		true

ii.

n	j	n % j	prime	isPrime(35)
9	2	1	true	
	3	0	false	
	4	1		
	5	4		
	6	3		
	7	2		
	8	1		false

iii. `for (int j = 2; j <= Math.sqrt(n); j++) {`

iv. replace Line 3 with:

```
3     j = 2;
3a    while (j <= Math.sqrt(double n) && prime)
```

After Line 5 add:

```
5a    j++;
```

b.

'C'	'O'	'M'	'P'	'U'	'T'	'E'	'R'
'S'	'C'	'I'	'E'	'N'	'C'	'E'	'X'
'I'	'S'	'X'	'F'	'U'	'N'	'X'	'X'

Question 6

a.

people	loc	pos	seats[pos]	findSeats(1)
3	1	1	'E'	
		2	'X'	false

Value returned is false.

b. The method would return true if the code had never returned false while executing the loop. This would mean none of the seats checked are taken, so all the tested seats are empty.

c. It would need to be passed as a parameter into the `findSeats` method. For example:

```
public static boolean findSeats(int loc, int people)
```

d. After line 24 add:

```
24a    if seats[loc]=='E' {
```

and include the line

```
27a    }
```

e.

people	answer	pos	seats[answer+pos]
2	1	0	seats[1] = 'X'
		1	seats[2] = 'X'

Final value in array seats:

'E'	'X'	'X'	'X'	'E'	'E'	'X'

Section C

Question 7

a.

i. `String`

ii. `print("hello ", "T")`

b.

i. `Monster dragon = new Monster("dragon", 72);`

ii.
```
Monster griffon = new Monster();
griffon.setName("griffon");
griffon.setHitPoints(65);
```

c.

i. `Rectangle rect1 = new Rectangle(100, 200, 300, 400);`

ii.
```
Rectangle rect2 = new Rectangle(200, 100, 400, 200);
Rectangle rect3 = new Rectangle(400, 200, 400, 200);
```

iii. `boolean b = rect2.intersects(rect3);`

Question 8

a.
```
FishMap fish1 = new FishMap("Flathead", -42.72, 148.05);
FishMap fish2 = new FishMap("Whiting", -42.88, 147.99);
```

b.
```
fish1.setLength(34);
fish2.setLength(27);
```

c. Distance to `fish1` is `fish1.distanceToFish(-42.82, 147.87);`

Distance to `fish2` is `fish2.distanceToFish(-42.82, 147.87);`

d. Distance from `fish1` to `fish2` is `fish1.distanceToFish(fish2.getLat(), fish2.getLon());`

e. 1 – Create a second constructor that takes the species and length as parameters.
2 – Create setter methods for latitude and longitude.
For example:

```
public FishMap(String species, int length) {
 this.species = species;
 this.length = length;
}

public void setLat(double la) {
 lat = la;
}

public void setLon(double lo) {
 lon = lo;
}
```

Question 9

a.

```
public class Innings {
 private String ground;
 private int runs;
 private int balls;
 private double strikeRate;
 private boolean out;

 public Innings(String g, int r, int b, boolean o) {
  ground = g;
  balls = b;
  out = o;
  calculateStrikeRate();
 }

 public boolean getOut() {
  return out;
 }

 public int getBallsFaced(){
  return balls;
 }

 public int getRunsScored() {
  return runs;
 }

 public String getGround() {
  return ground;
 }

 public void calculateStrikeRate() {
  if (balls == 0) {
   strikeRate = 0;
  } else {
   strikeRate = (double) runs / balls * 100;
 }

 public double getStrikeRate() {
  return strikeRate;
 }
}
```

b.

```
System.out.println("Career of Ricky Warne at " + ground + ":")
int count = 0;
String output;

for (int i = 0; i < 5; i++) {
 if (career[i].getGround().equals(ground)) {
  count++;
  output = count + ": ";
  output += ground + ", ");
  output += "SR "+ career[i].getStrikeRate();
  output += ", out? ";
  if (career[i].getOut()) {
   output += "Yes";
  } else {
   output += "No";
  }
  System.out.println(output);
 }
}
```

Section D

Question 10

a.

i. "Program Counter"

ii. $C \equiv \sim(A \wedge B)$

iii. $\sim A$

iv.

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

b. $D \equiv (A \wedge B) \vee (\sim A \wedge \sim C)$

c.

i.

A	B	$\sim A$	$\sim B$	$\sim A \vee \sim B$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

A	B	$A \vee B$	$\sim(A \vee B)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

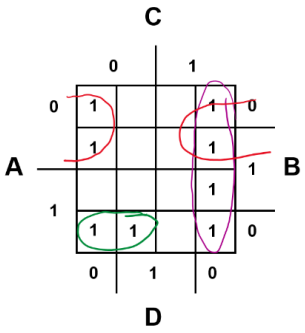
ii. The last column of each truth table is different, therefore $\sim A \vee \sim B$ is not equivalent to $\sim(A \vee B)$.

Question 11

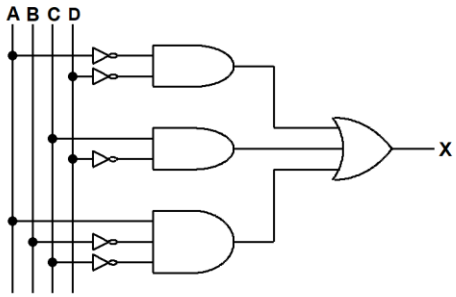
a. TOY programs begin execution from line 10. By convention, the contents of memory locations 00 to 0F are considered data and not program code.

b.

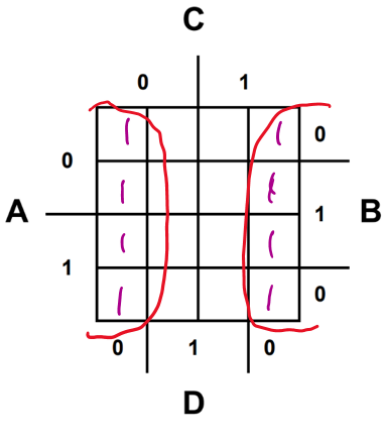
i.



$(\sim A \wedge \sim D) \vee (C \wedge \sim D) \vee (A \wedge \sim B \wedge \sim C)$



ii.



$Y \equiv \sim D$

c.

$E \equiv (\sim(\sim A \wedge B) \wedge A) \vee ((A \vee \sim B) \wedge \sim A)$

$\equiv \big((\sim\sim A \vee \sim B) \wedge A\big) \vee ((A \vee \sim B) \wedge \sim A)$ L11

$\equiv \big((A \vee \sim B) \wedge A\big) \vee ((A \vee \sim B) \wedge \sim A)$ L6

$\equiv \big((A \vee \sim B) \wedge A\big) \vee (\sim A \wedge \sim B)$ L28

$\equiv (A) \vee (\sim A \wedge \sim B)$ L33

$\equiv A \vee \sim B$ L27

Question 12

a.

i.

R[A]	1	0	1	0	0	0	1	1	0	1	0	1	1	0	0	1
R[B]	0	0	1	1	1	1	0	0	1	0	0	1	0	1	0	1
R[C]	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1

ii.

PC	R[1]	R[A] in binary	R[B]	R[C] in binary
10		1101		
11	0001			
13				0001
15			1	
16		0110		
13				0001
15			2	
16		0011		
13				0000
16		0001		
13				0001
15			3	

iii. The purpose of the program is to count the number of 1s in the user input.

b.

Memory Address	**Value**	**Pseudocode**
01	0001	
02	0009	
10	8A01	R[A] ← mem[01]
11	8B01	R[B] ← mem[01]
12	8F02	R[F] ← mem[02]
13	8101	R[1] ← mem[01]
14	2FF1	R[F] ← R[F] – R[1]
15	DF18	If (R[F] > 0) goto 18
16	9AFF	write R[A]
17	0000	Halt
18	1CAB	R[C] ← R[A] + R[B]
19	1AB0	R[A] ← R[B]
1A	1BC0	R[B] ← R[C]
1B	C014	goto 14

The 9th Fibonacci number is $22_{16} = 34_{10}$.

Section E

Question 13

a.

i. $0101\ 1010_2 = 5A_{16}$

ii. $73_{10} = 0100\ 1001_2$

iii.

```
  1 1 0 1 1
+ 1 1 0 1
-----------
1 1 0 0 0
```

b.

i. = 128 + 64 + 4 + 2 + 1 = 199

ii.

```
  1100 0111
+ 0001 0000
-----------
  1101 0111
```

iii. 199 + 16 = 215
Also, 128 + 64 + 16 + 4 + 2 + 1 = 215

iv. 1001 1101 in 8-bit two's complement.
0110 0011 = 64 + 32 + 2 + 1 = 99.

So 1001 1101 = -99 in 8-bit twos complement.

v. $0111\ 1111_2 = 127_{10}$

Question 14

a. dice has 6 values, so 3 bits are needed to represent the values.

b.

i. 1 00010 110000000
sign = 1 → negative
exponent = 00010 → + 00010 = +2
mantissa = 0.11 → 0.5 + 0.25 = 0.75

decimal value = - 0.75 × 2^2 = -3

ii. negative, so sign = 1
mantissa = 0.25 = 010000000
exponent = 1 = 00001

1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

iii. So long as the mantissa is 0, the other 6 bits can have any value and the number represented will “look like” 0.
2^6 = 64, so there are 64 values that can be interpreted as 0.

c. $44.1 \times 1000 \times 16 \times 45 = 31\,752\,000$

Question 15

a. For each subject:
subject code: 10x16 = 160 bits
4 bits per criteria: 4x10 = 40 bits
text comment: 1000x16 = 16 000 bits
parent teacher: 1 boolean value = 1 bit
Total = 16201 bits

For each report:
4 subjects: 16201x4 = 64 804 bits
photo: 24x200x300 = 1 440 000 bits
name: 256x16 = 4 096 bits
Total for each report = 1 508 900 bits

For 1000 students:
1000 x 1508900 = 1 508 900 000 bits (= 188 612 500 bytes or ~180 MB)

b.

i. 3 bits to represent the 8 block types.
3x64x64x16 = 196 608 bits

ii. For each tower:
5 bits to store the number of blocks in that tower (to store values 0 to 16 inclusive)
3 bits for each block = 3x5 = 15 bits.
Total: 20 bits for each tower
Word total:
64x64x20 = 81 920
A saving of 196 608 – 81 920 bits = 114 688 bits

iii. The compression is lossless since no data about the world is lost when the data is compressed and decompressed. Note that the compression would not actually be smaller if all the towers were 15 blocks high.