Mark Scheme (Results)

Summer 2025

Pearson Edexcel International GCSE In Computer Science (4CP0)

Paper 2C: Application of Computational Thinking

Edexcel and BTEC Qualifications

Edexcel and BTEC qualifications are awarded by Pearson, the UK's largest awarding body. We provide a wide range of qualifications including academic, vocational, occupational and specific programmes for employers. For further information visit our qualifications websites at www.edexcel.com or www.btec.co.uk. Alternatively, you can get in touch with us using the details on our contact us page at www.edexcel.com/contactus.

Pearson: helping people progress, everywhere

Pearson aspires to be the world's leading learning company. Our aim is to help everyone progress in their lives through education. We believe in every kind of learning, for all kinds of people, wherever they are in the world. We've been involved in education for over 150 years, and by working across 70 countries, in 100 languages, we have built an international reputation for our commitment to high standards and raising achievement through innovation in education. Find out more about how we can help you and your students at: www.pearson.com/uk

Summer 2025
Question Paper Log Number P83974A
Publications Code 4CP0_2C_2506_MS
All the material in this publication is copyright
© Pearson Education Ltd 2025

General Marking Guidance

- All candidates must receive the same treatment. Examiners must mark the first candidate in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Candidates must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- There is no ceiling on achievement. All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved, i.e. if the answer matches the mark scheme. Examiners should also be prepared to award zero marks if the candidate's response is not worthy of credit according to the mark scheme.
- Where some judgement is required, mark schemes will provide the principles by which marks will be awarded and exemplification may be limited.
- When examiners are in doubt regarding the application of the mark scheme to a candidate's response, the team leader must be consulted.
- Crossed out work should be marked UNLESS the candidate has replaced it with an alternative response.

Mark Scheme – Java

Question	mp	Answer	Additional Guidance	Mark
1(a)	A1	**The only correct answer is B** ***A*** *is not correct because for is used in iteration* ***C*** *is not correct because new is used to declare an object* ***D*** *is not correct because while is used in repetition*		**1**

Question	mp	Answer	Additional Guidance	Mark
1(b)(i)	B1 B2 B3	Award one mark for each correct cell: **Description** / **Syntax error** / **Logic error** / **Runtime error** The program does not produce the expected output. / / ✔ / The program does not translate. / ✔ / / The program crashes during execution. / / / ✔		**3**

Question	mp	Answer	Additional guidance	Mark
1(b)(ii)	B4	Line 19 – { added to end of line		
	B5	Line 25 – 0 changed to 1		
	B6	Line 29 missing + added between string and greenCount		3

```
1  // Q01bii
2
3  public class Q01bii
4  {
5      public static void main (String[] args)
6      {
7          // initialise counters
8          int ballCount = 0;
9          int greenCount = 0;
10         int redCount = 0;
11
12         String[] ballArray = {"green", "red", "green", "green", "red", "green", "green", "green", "green", "green",
13         "green", "red", "green", "red", "green", "red", "red", "green", "green", "red", "red", "green", "green",
14         "red", "green", "red", "red", "green", "green", "red", "green", "green", "red", "green", "green", "green",
15         "red", "green", "red", "green"};
16
17         // iterate over ball array
18
19         for (String ball : ballArray) {
20             ballCount += 1;
21             if(ball == "green"){
22                 greenCount += 1;
23             }
24             else {
25                 redCount += 1;
26             }
27
28         }
29         System.out.println("Green: " + greenCount + " balls \n" + "Red: " + redCount + " balls \n" + "Total: " + ballCount + " balls");
30
31     }
32 }
```

Question	mp	Answer	Additional guidance	Mark
1(c)(i)	C1	Award one mark for: • To store a value that must not be changed/prevent accidental change of value • To simplify code maintenance/so that if the value must be updated the change only has to made in a single place • To avoid inconsistency error of specifying same value multiple times • To make code more readable/give meaningful names to values	Accidental change cannot refer to 'user'	**1**

Question	mp	Answer	Additional guidance	Mark
1(c)(ii)	C2 C3	Award **two** marks for a linked description such as: • A variable stores a single value (1) whereas an array stores multiple values (1)		**2**

Question	mp	Answer	Additional guidance	Mark
1(d)(i)	D1	Award **one** mark for: • getWages (1)	Accept 'main' Accept: `Static double getWages(double inSales, int inAge)` line 5 (C#)	**1**

Question	mp	Answer	Additional guidance	Mark
1(d)(ii)	D2	Award **one** mark for: • && (1)	Do not accept line numbers Accept: and	**1**

Question	mp	Answer	Additional guidance	Mark
1(d)(iii)	D3	Award **one** mark for any of the following: • sales (1) • wages (1) • inSales (1)	Do not accept line numbers	**1**

Question	mp	Answer	Additional guidance	Mark
1(d)(iv)	D4	Award **one** mark for any of the following: • inSales (1) • inAge (1)	Do not accept line numbers	**1**

Question	mp	Answer	Additional Guidance	Mark
2(a)	A1	**The only correct answer is D** ***A*** *is not correct because abstraction is a technique used in computational thinking* ***B*** *is not correct because encryption is a security measure* ***C*** *is not correct because iteration is a programming construct*		**1**

Question	mp	Answer	Additional guidance	Mark
2(b)(i)		Award **one** mark for each correct cell:		
	B1 B2	real/double/float: (1) • pay rate/18.75 (1) • total pay/356.25 (1) • tax/0.2/71.25 (1) • income/285.00 (1) int/integer: (1) • hours/19 (1)	Example must match data type for B2 Award suitable example even if not taken from payslip Accept string and matching example e.g. Hours	**2**

Question	mp	Answer	Additional guidance	Mark
2(b)(ii)		Award **one** mark for each correct cell:		
	B3	**Input:** • Pay Rate/18.75 (1) • Hours/19 (1)		
	B4	**Process**: • Pay Rate * Hours/18.75 * 19 (1) • Total Pay * Tax/356.25 * 0.2 (1) • Total Pay – Tax/356.25 – 71.25 (1)	Don't need output e.g. total pay, just the calculation part e.g. pay rate * hours: Total pay = pay rate * hours Accept 0.2 or 20% as part of calculation for tax Ok to mix types e.g. Total Pay * 20% Accept descriptions such as *multiply the pay rate by hours (hours etc could be taken from input box/ B3 mark)*	
	B5	**Output:** • Total Pay/356.25 (1) • Tax/71.25 (1) • Income/285 (1)		**3**

Question	mp	Answer	Additional Guidance	Mark
2(c)(i)	C1 C2	Award **one** mark for **one** correct cell: Award **two** marks for all **three** correct cells: • **Normal:** any value between 0-255 e.g. 125 • **Boundary:** 0/255 • **Erroneous:** any negative value/any value 256 or above	Accept -1/256 for boundary data Don't accept non-int types	**2**

Question	mp	Answer	Additional Guidance	Mark
2(c)(ii)		Award **one** mark for any of the following up to a maximum of **four** marks:		
	C3	Input changed to int before comparison (1)		
	C4	Relational operator used correctly to check for valid input (both **>=**0 and **<**256 or equivalent) (1)	Allow: >0 and/or <255	
	C5	'&&' logical operator used to combine relational tests (1)	or operator acceptable if candidate rearranges the selection statement and test works for correct code option – check if C4 is now awardable	
	C6	Calling of `getHex(userInput)` (1)	Call must be inside print ( ) Subprogram and argument must be functionally correct Ignore additional variables/strings in print statement	**4**

```
42      public static void main(String[] args)
43      {
44          int userInput = 0;
45
46          Scanner myKeyboard = new Scanner (System.in);
47
48          System.out.println ("please enter a number between 0 and 255 \n");
49
50          // complete the line to take the user input as an integer
51          userInput = myKeyboard.nextInt();
52
53          // complete the test to check the input is between 0 and 255
54          if((userInput < 0) || (userInput > 255))
55          {
56              System.out.println("number out of range");
57          }
58          else
59          {
60              // complete the line to call getHex subprogram
61              System.out.println(getHex(userInput));
62          }
63      }
64  }
```

<table>
<tr><th>Question</th><th>mp</th><th>Answer</th><th>Additional Guidance</th><th>Mark</th></tr>
<tr>
<td>3(a)(i)</td>
<td>A1
A2</td>
<td>Award one mark for each:
• Pass 1 correct (1)
• Pass 2 correct (1)
<table>
<tr><td>9</td><td>11</td><td>12</td><td>19</td><td>14</td><td>18</td><td>15</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>18</td><td>15</td><td>17</td><td>19</td><td>(1)</td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>15</td><td>17</td><td>18</td><td>19</td><td>(1)</td></tr>
</table>

Must show ONLY the 19 moving in first pass and ONLY the 18 moving in second pass</td>
<td>Ignore lines with individual swaps and only award correct completed passes
<table>
<tr><td>9</td><td>11</td><td>12</td><td>19</td><td>14</td><td>18</td><td>15</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>19</td><td>18</td><td>15</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>18</td><td>19</td><td>15</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>18</td><td>15</td><td>19</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>18</td><td>15</td><td>17</td><td>19</td><td>(1)</td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>15</td><td>18</td><td>17</td><td>19</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>15</td><td>17</td><td>18</td><td>19</td><td>(1)</td></tr>
</table>

Accept accurate right-to-left bubble sort as shown below
<table>
<tr><td>9</td><td>11</td><td>12</td><td>19</td><td>14</td><td>18</td><td>15</td><td>17</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>19</td><td>15</td><td>18</td><td>17</td><td>(1)</td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>15</td><td>19</td><td>17</td><td>18</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>15</td><td>17</td><td>19</td><td>18</td><td></td></tr>
<tr><td>9</td><td>11</td><td>12</td><td>14</td><td>18</td><td>15</td><td>17</td><td>19</td><td>(1)</td></tr>
</table>

Complete sort cannot be the 1st /only line in response – 0 marks as this will be taken as both 18 and 19 moving in first pass</td>
<td>2</td>
</tr>
</table>

Question	mp	Answer	Additional Guidance	Mark
3(a)(ii)	A3	Award **one** mark for: • 4	If right-to-left pass completed in Q3ai, can only accept 2 swaps	**1**

Question	mp	Answer	Additional Guidance	Mark
3(a)(iii)	A4	Award **one** mark for: • 3	Do not accept 2 – 3rd pass required for **algorithm** to finish If right-to-left pass completed in Q3ai, can only accept 5 passes	**1**

Question	mp	Answer	Additional guidance	Mark
3(b)	B1	Award **one** mark for: • Binary search	Accept any other divide and conquer search algorithm (google if necessary) Accept 'binary' on its own as search in question	**2**
	B2	Award **one** mark for any of the following: • Divide and conquer / list halves each time(1) • Fewer searches needed (1)	Do not accept points about being 'already sorted' as this is in question	

Question	mp	Answer	Additional guidance	Mark
3(c)		Award **one** mark for each of:		
	C1	(while) Loop ends **only** when 0 entered (1)	Any method e.g. `while (age != 0):`	**5**

	C2	Avoiding divide by zero error (1)	By coding 2nd selection box from flowchart – no output required
			Must come after some attempt to process age or count – award even if inside loop.
			ignore 'other' divide by zero errors introduced by candidate
			Must follow logic of flowchart e.g.:
			`if (count == 0){ (or count != 0)`
			…
			`else`
			…	

	C3	Average value rounded to whole number (1)	Must come after some attempt to process age or count – award even if inside loop. Must target calculated average (even if formula not correct) Any method to get int Accept casting to int *(round not in spec, but string manipulation is)* Accept integer division	

	C4	Correct value calculated for average and count (1)	Allow even if 0 used to initialise count in B1 (this must be only error in calculation) – set count to -1 to test	
	C5	Count and average included in a syntactically correct final print statement (1)	Award even if never runs Must include both variables (may also include string)	

```java
// Q03c

import java.lang.Math;
import java.util.Scanner;

public class Q03c
{
    Run | Debug
    public static void main(String[] args)
    {
        // program variables
        int age = 1;
        int total = 0;
        int average = 0;
        int count = -1;
        // ---------------------------------
        // Write your code below this line
        while (age !=0)
        {
            Scanner myKeyboard = new Scanner (System.in);

            System.out.println ("Please enter age in years, (or enter 0 to finsh)");
            age = myKeyboard.nextInt();
            // increment total and count
            total += age;
            count += 1;

        }
        if(count == 0){
            System.out.print("No ages entered");
        }
        else {
        // calculate average and round to whole number
        average = Math.round(total/count);

        // output count and average
        System.out.printf("You entered %d member ages, with an average age of %d years.", count, average);
        }
    }
}
```

Question	mp	Answer	Additional Guidance	Mark
4(a)	A1	Breaking problem into small parts		**1**

Question	mp	Answer	Additional guidance	Mark
4(b)		Award **two** marks for a linked explanation such as:		
	B1 B2	• Programmer can ignore unnecessary detail (1) because it doesn't matter how library generates the number (1) • Eli can focus on just his own code (1) because library hides the code that generates the random numbers (1) • It doesn't matter how the subprogram generates the random number (1) because Eli only needs the number itself (1)	First mark for meaning of abstraction (focus on necessary parts / hiding unnecessary detail) Second mark for linking use of **random** library Do **not** accept answers relating to modelling real-world scenarios as this is in the question	**2**

Question	mp	Answer	Additional guidance	Mark
4(c)		Award **one** mark for each of:		
	C1	Booster variable initialised to 1 (1)	booster = 1	
	C2	At least one given constant used correctly (1)		
	C3	Loop used to make three throws (1)	Either type	
	C4	Use of random.nextInt( ) to assign roll a value between 1 and 6 (1)	roll = random.nextInt(SIDES) +1 roll = random.nextInt(7) Any accurate method	

	C5	Check for both 3 or 6 for at least one roll (1)	any method e.g.
			`if (roll == 3 || roll == 6)`
			`if roll % 3 == 0`	

	C6	booster incremented (1)	Must be indented under condition in C5. Can award even if C5 not functioning correctly, but must be idea of only boosting on 3 and 6	
	C7	Correct final score calculated from 3 random rolls (1)	Even if not displayed Does not need to use given variable (finalScore) to get this mark	
	C8	Correct output strings including each roll score and final score with appropriate strings (1)	All three throws and final score as shown in figure 6. `roll` and `finalScore` variables should match candidate solution Allow sensible alternate strings e.g. Roll for Rolled or Total for Score – Ignore : (colon)	**8**

```java
// Q04c

import java.util.Scanner;
import java.util.Random;

public class Q04c
{
    // program constants
    public static final int SIDES = 6;
    public static final int THROWS = 3;

    Run | Debug
    public static void main(String[] args)
    {
        // complete the lines to initialise the variables
        int roll = 0;
        int subtotal = 0;
        int booster = 1;
        int finalScore = 0;

        // roll dice 3 times
        for(int thro = 1; thro <= THROWS; ++thro)
        {
            Random random = new Random();
            roll = random.nextInt(SIDES) + 1;
            // display number rolled
            System.out.println("Rolled " + thro + ": " + roll);

            // update subtotal
            subtotal = subtotal + roll;

            // check for roll of 3 or 6 and increment booster if needed
            if(roll == 3 || roll == 6)
            {
                booster += 1;
            }
        }
        // Calculate and output final score
        finalScore = subtotal * booster;
        System.out.println("Score: " + finalScore);
    }
}
```

Question	mp	Answer	Additional guidance	Mark
5(a)(i)	A1 A2	final score not accurate (1) because score is resetting/score keeps going back to zero (in each iteration of the loop) (1)	'Score not accurate' is not enough on its own 'Score on line 20' or 'Score output to DISPLAY' would count as final score.	**2**

Question	mp	Answer	Additional guidance	Mark
5(a)(ii)	A3 A4	Award up to **two** marks for a linked description such as: • Asking 10 addition questions (1) and calculating final score (1) • Asking randomised adding questions (1) and checking user answer against computer answer (1) • Creating adding quiz with random numbers up to 99 (1) and outputting number of correct answers (1)	Processes that can be included Concept of this being an 'Adding Quiz' required • Getting random numbers for question • Calculating sum of two random numbers for correct answers • comparing user answer to correct answer • incrementing /tracking score • asking 10 questions (loop) • outputting final score	**2**

Question	mp	Answer	Additional guidance	Mark
5(b)				
	B1	Open file for writing (1)		
	B2	Selection used to get grade (1)	Must address all 4 grades Does not have to be accurate on boundaries e.g. >80 is ok for distinction etc.1	
	B3	Output suitable message to say file created (1)		
	B4	Evidence of 2D indexing (1)	Even if element not used in rest of program 2D indexing **or** 1D index within *for-each* structure **or** sum(1Darray) or min(1Darray) etc. At least 1 inner value accessed	
	B5	Lowest score **must** be found and omitted/subtracted from calculated scores (all arrays) (1)	Any method e.g.: • List methods • Selection statements • iterable functions must be done programmatically	
	B6	At least 1 line written to file (1)	Must contain something generated by program (total/result), does not have to be correct total or grade	
	B7	Add line feed to write line after both score and grade (1)		
	B8	Output file all correct (1)		**8**

Paper 2 – Application of Computational Thinking

```java
// Q05b

import java.io.IOException;
import java.util.Scanner;
import java.io.FileWriter;
import java.util.Arrays;

public class Q05b
{
    Run | Debug
    public static void main(String[] args) throws IOException {
        int[][] Tbl_scores = {{45,42,43},{19,36,41},{48,48,48},
        {35,32,31},{18,22,26},{41,35,31},{15,19,12},{41,39,40},
        {27,25,22},{32,36,35},{37,36,37},{40,35,41}};

        // ---------------------------------------------
        // Write your code below this line

        // program variables
        int total = 0;
        String result = "";
        String outString = "";

        //open file for writing
        FileWriter writer = new FileWriter("grades.txt");

        // sort score arrays and add highest two scores
        for (int [] score : Tbl_scores)
        {
            Arrays.sort(score);
            total = score[1] + score[2];

            // decide result based on total
            if(total >= 80) {result = "Distinction";}
            else if(total >= 65) {result = "Merit";}
            else if(total >=50) {result = "Pass";}
            else {result = "Fail";}

            // write total and result to file
            outString = total + " " + result + "\n";
            writer.write(outString);
```

```
41
42          };
43
44          // confirm gradeds file created
45          System.out.println("Grades file created");
46
47          // Close the writer to release resources and ensure data is written
48          writer.close();
49      }
50  }
```

Question	mp	Answer	Additional guidance	Mark
6		Award **one** mark for each of:		
	A1	Both City and Temp taken as input (1)		
	A2	Both correct conversion calculation coded (1)	Does not have to be used	

	A3	Any relevant subprogram has been declared (1)	At least one **relevant** subprogram declared.
			`public static <type> name(param):`
			or
			`public static <type> name():`
			and one line of code indented	

	A4	Temperature in correct parallel position is updated when existing city is input (1)		
	A5	Loop for accessing all items in either array (1)	**For each** acceptable for this mark	
	A6	Table includes header and dividing line (1)	`Console.WriteLine("city Celsius fahrenheit");` `Console.WriteLine("-------------");`	
	A7	Mechanism for getting temp from string and completing conversion (1)	Include int( ) and passed to formula for at least one temperature value	
	A8	Mechanism for getting scale C/F (1)		
	A9	All rows output to table with corresponding city and (at least one) temperature values (1)	(same index value used) do not accept hard coded	
	A10	Table (for cities) all correct (1)	Ignore header (A6) and average (A11) Must have city, C value and F value (all accurate) Ignore float values if calculation correct	**11**

	A11	Overall average correctly calculated and displayed in both scales (1)	Average may change depending on whether whole number cast or rounded – award accurate formula, not result.	

LBMS (AOs: 3xAO3.2a, 3xAO3.2b, 3xAO3.3c) (Tgt: 3, 3, 3); Spec: 1.1.6(2), 1.2.1(2), 1.2.2(2), 2.1.1, 2.1.2, 2.1.5
PBMS: 2.1.1, 2.3.2(2), 2.3.3(2), 2.4.1(2), 2.5.1(2), 2.6.1(2)

Award up to a maximum of nine marks using the levels-based mark scheme below.				Mark
Band 0	**Band 1 (1-3 marks)**	**Band 2 (4-6 marks)**	**Band 3 (7-9 marks)**	
No rewardable content	Little attempt to decompose the problem into component parts	Some attempt to decompose the problem into component parts	The problem has been decomposed into component parts	
	Some parts of the logic are clear and appropriate to the problem	Most parts of the logic are clear and mostly appropriate to the problem	The logic is clear and appropriate to the problem	
	Some appropriate use and manipulation of data types, variables, data structures and program constructs	The use and manipulation of data types, variables and data structures and program constructs is mostly appropriate	The use and manipulation of data types, variables and data structures and program constructs is appropriate	
	Parts of the code are clear and readable	Code is mostly clear and readable	Code is clear and readable	
	Finished program will not be flexible enough with other data sets or input	Finished program will function with some but not all other data sets or input	Finished program could be used with other data sets or input	
	The program meets some of the given requirements	The program meets most of the given requirements	The program fully meets the given requirements	**(9)**

```java
// Q06

import java.lang.Math;
import java.util.Arrays;
import java.util.Scanner;

public class Q06
{
    // function to convert centigrade to fahrenheit
    public static int convertCtoF(int tempC)
    {
        double tempFd = (1.8 * tempC) + 32;
        int tempFi = (int)Math.round(tempFd);
        return tempFi;
    }

    // function to convert fahrenheit to centigrade
    public static int convertFtoC(int tempF)
    {
        double tempCd = (tempF - 32) / 1.8;
        int tempCi = (int)Math.round(tempCd);
        return tempCi;
    }

    Run | Debug
    public static void main(String[] args)
    {
        // Average temperatures for citys
        String[] Tbl_city = { "Reggane", "Cairo", "New Delhi", "Muscat", "Athens", "Barcelona", "Havana", "Phoenix", "Brisbane", "Darwin" };
        String[] Tbl_temps = { "F103", "F82", "C34", "F95", "C29", "F79", "F84", "C35", "F77", "C27" };

        // ------------------------------------------------
        // Write your code below this line

        // Variables
        String city = "";
        String inputCity = "";
        String inputTemp = "";
        String temp = "";
        char tempType = 'x';
        int arrayLength = Tbl_city.length;
```

```
41          int tempValue = 0;
42          int thisCtemp = 0;
43          int thisFtemp = 0;
44          boolean inlist = false;
45          int subtotalTemp = 0;
46          int averageTemp = 0;
47
48          Scanner myKeyboard = new Scanner (System.in);
49
50          // update city temperature
51          System.out.println ("Enter the city name");
52          inputCity = myKeyboard.nextLine();
53
54          for (int i = 0; i < arrayLength; i++)
55          {
56              // System.out.println(i);
57              if(Tbl_city[i].equals(inputCity))
58              {
59                  System.out.println ("Enter temperature (include C/F) ");
60                  inputTemp = myKeyboard.nextLine();
61                  Tbl_temps[i] = inputTemp;
62                  inlist = true;
63              }
64          }
65          if(inlist == false) {
66              System.out.println("City not found");
67          }
68
69          // Print the Table Header
70          String Layout = "%-15s %12s %12s %n";
71          System.out.printf(Layout, "City", "Celsius", "Fahrenheit");
72          System.out.println("-----------------------------------------");
73
74
75          // Print each row in the table
76          for(int i=0; i < arrayLength; i++)
77          {
78              city = Tbl_city[i];
79              temp = Tbl_temps[i];
80              tempType = temp.charAt(0);
```

```
81              tempValue = Integer.valueOf(temp.substring(1));
82
83              // decide which conversion to do
84              if(tempType == 'F' || tempType == 'f')
85              {
86                  thisCtemp = convertFtoC(tempValue);
87                  thisFtemp = tempValue;
88              }
89              else
90              {
91                  thisFtemp = convertCtoF(tempValue);
92                  thisCtemp = tempValue;
93              }
94
95              System.out.printf(Layout, city, thisCtemp, thisFtemp);
96
97              // tracking average
98              subtotalTemp = subtotalTemp + thisCtemp;
99          }
100         // display average line
101         averageTemp = (int)(subtotalTemp / arrayLength);
102         System.out.println("-----------------------------------------");
103         System.out.printf(Layout, "Overall Average", averageTemp, convertCtoF(averageTemp));
104     }
105 }
```

Pearson Education Limited. Registered company number 872828
with its registered office at 80 Strand, London, WC2R 0RL, United Kingdom