Pearson
Edexcel

# Mark Scheme (Provisional)

Summer 2021

## Pearson Edexcel International GCSE
## in Computer Science (4CP0_2C)
## Paper 02: Application of Computational Thinking - Java

Summer 2021
Question Paper Log Number P66490A
Publications Code 4CP0_2C_2106_MS

## 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.

## Theory Mark Scheme

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (a) | A1 | **The only correct answer is B**<br>*A is not correct because as it is an arithmetic operator*<br>*C is not correct because as it is a relational operator*<br>*D is not correct because as it is a relational operator* | | (1) |

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (b) | B1<br>B2 | Award up to 2 marks for a linked description such as:<br>• 1D represents items as a list (1), 2D as a table (1)<br>• 1D is a row (1), 2D is a table (1)<br>• Each element in 1D is a single value (1), each element in 2D is a 1D array (1) | Ignore capitalisation | (2) |

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 2 (c) | | Award 1 mark for each set of test data. | | (3) |

| mp | | Test data | Expected results |
|---|---|---|---|
| C1<br>C2<br>C3 | booksSold<br>profit | Either of<br>• booksSold = 4<br>• profit = 4 | Poor performances this week |
| | booksSold<br>profit | 5<br>10 | Sales and profit are good this week |
| | booksSold<br>profit | 21<br>20 | Sales and profit are excellent this week |

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 3 (b) | B1 | Award up to 2 marks for a linked explanation such as:<br>• The number of keys are limited (1) making it easy to use brute force to decrypt (1)<br>• It can be easy to find commonly used letters (e.g. E) (1) and guess the key (1) | Accept alternative similar wording. | (2) |

<table>
<tr><th>Question</th><th>mp</th><th>Answer</th><th>Additional Guidance</th><th>Mark</th></tr>
<tr><td rowspan="5">3 (c)</td><td></td><td>Award 1 mark each up to a maximum of 4 for:
<table>
<tr><td>Encrypted letter</td><td>f</td><td>l</td><td>m</td><td>k</td><td>t</td><td>r</td><td>w</td><td>h</td><td>e</td><td>e</td></tr>
<tr><td>Keyword letter</td><td>t</td><td>h</td><td>i</td><td>r</td><td>t</td><td>y</td><td>t</td><td>h</td><td>i</td><td>r</td></tr>
<tr><td>Decrypted letter</td><td>m</td><td>e</td><td>e</td><td>t</td><td>a</td><td>t</td><td>d</td><td>a</td><td>w</td><td>n</td></tr>
</table></td><td></td><td rowspan="5">(4)</td></tr>
<tr><td>C1</td><td>Ciphertext mapped to keyword in row 2 (1)</td><td></td></tr>
<tr><td>C2</td><td>At least one letter decrypted correctly (1)</td><td></td></tr>
<tr><td>C3</td><td>At least one word decrypted correctly (1)</td><td></td></tr>
<tr><td>C4</td><td>Decrypted message ‘meet at dawn’ (1)</td><td></td></tr>
</table>

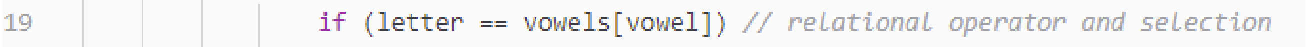
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 3 (d)(i) | D1 | Award 1 mark for:<br>• cipherLetter / a single encrypted letter (1) | Do not accept word/message/text | (1) |
| 3 (d)(ii) | D2 | Award 1 mark for any of:<br>• keywordLetter<br>• plaintextLetter | Ignore case | (1) |
| 3 (d)(iii) | D2 | Award 1 mark for any of:<br>• subprogram that is already defined<br>• subprogram that is already written<br>• subprogram that is already compiled<br>• subprogram that can be called without having to write code for it | | (1) |

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 4 (b)(i) | B1 | Award up to 2 marks for a linked explanation:<br>• binary search can be quicker than a linear search (1) as it does not have to examine each item in the list (1)<br>• binary search halves the list each time (1) so it can be faster to find an item (1)<br>• binary search requires fewer comparisons than a linear search to establish an item is not in the list (1) because the linear list would need to compare each item before establishing this (1) | | (2) |

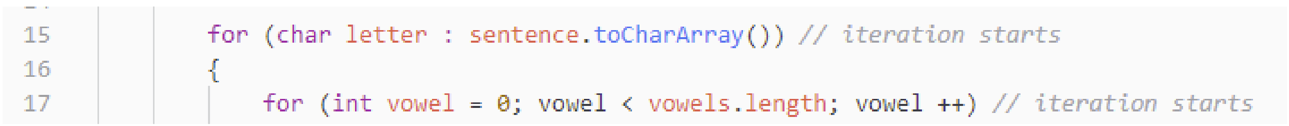
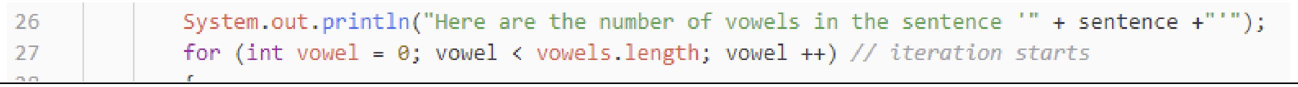
<table>
<tr>
<td rowspan="5">4 (b)(ii)</td>
<td colspan="2">Correct answer
<table>
<tr><th>Position in list</th><th>Product code</th><th>Order examined</th></tr>
<tr><td>1</td><td>ark11</td><td></td></tr>
<tr><td>2</td><td>asp11</td><td></td></tr>
<tr><td>3</td><td>bar13</td><td></td></tr>
<tr><td>4</td><td>dri15</td><td>1</td></tr>
<tr><td>5</td><td>mil19</td><td></td></tr>
<tr><td>6</td><td>rib10</td><td>2</td></tr>
<tr><td>7</td><td>str15</td><td>3</td></tr>
<tr><td>8</td><td>tor16</td><td></td></tr>
</table>
Award one mark for each correct value in order column</td>
<td></td>
<td>(4)</td>
</tr>
<tr>
<td>B2</td>
<td>Start of search correct</td>
<td>Accept 5 and 7 for B2 and B3 (2 marks)</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>B3</td>
<td>Second search item correct</td>
<td>Follow through if start of search incorrect</td>
</tr>
<tr>
<td>B4</td>
<td>Third search item correct</td>
<td>Follow through if start of search incorrect</td>
</tr>
<tr>
<td>B5</td>
<td>All correct</td>
<td></td>
</tr>
<tr>
<td>4 (b)(iii)</td>
<td>B6</td>
<td>Award 1 mark for:<br>3 **or** $\log_2 n + 1$</td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>4 (b)(iv)</td>
<td>B7</td>
<td>Award 1 mark for any of:<ul><li>bubble sort</li><li>merge sort</li></ul></td>
<td>Accept any known sorting algorithm</td>
<td>(1)</td>
</tr>
</table>

# Java Code Mark Scheme

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (c) | C1 | Change num_twenties == to num_twenties = (1) | | |
| | C2 | The **left over** variable named the same in both places (1) | | |
| | C3 | Change , to + | | (3) |

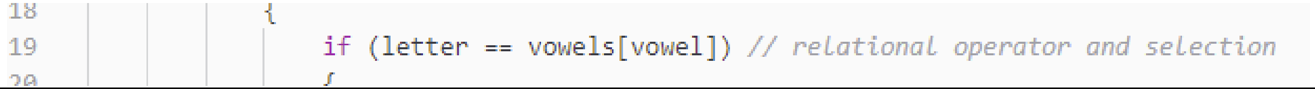
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (d)(i) | D1 | Award 1 mark for adding a comment at the end of the line where there is relational operator:  | May be on different line numbers | (1) |

```java
            if (letter == vowels[vowel]) // relational operator and selection
```

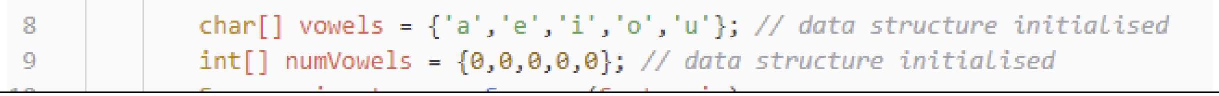
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (d)(ii) | D2 | Award one mark for adding a comment at the end of a line where iteration starts:   | May be on different line numbers | (1) |

```java
        for (char letter : sentence.toCharArray()) // iteration starts
        {
            for (int vowel = 0; vowel < vowels.length; vowel ++) // iteration starts
```

```java
    System.out.println("Here are the number of vowels in the sentence '" + sentence +"'");
    for (int vowel = 0; vowel < vowels.length; vowel ++) // iteration starts
```

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (d)(iii) | D3 | Award one mark for adding a comment at the end of the line where selection starts:  | May be on different line numbers | (1) |

```java
        {
            if (letter == vowels[vowel]) // relational operator and selection
```

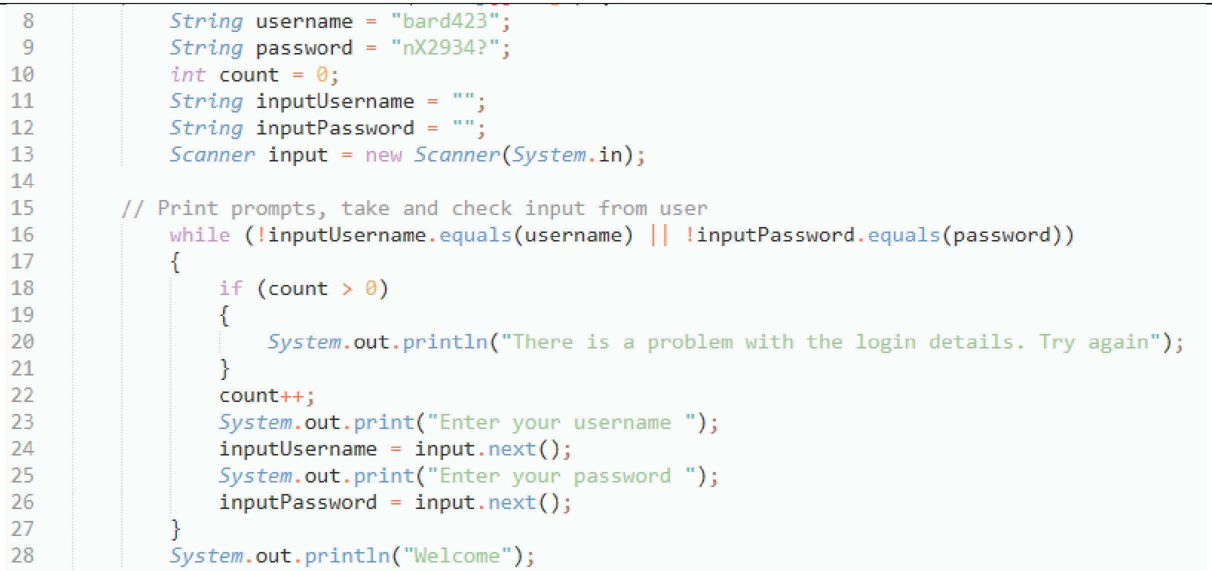
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 1 (d)(iv) | D4 | Award one mark for adding a comment at the end of a line where a data structure is initialised:  | May be on different line numbers | (1) |

```java
    // initialise variables
    char[] vowels = {'a','e','i','o','u'}; // data structure initialised
    int[] numVowels = {0,0,0,0,0}; // data structure initialised
```

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 2 (a) | | Award one mark for each of: | Logic of algorithm must be followed as set out.<br>Alternatives must address each point.<br>Do not penalise candidates who attempt more than the stated requirements.<br>Don’t penalise spelling mistakes and alternative wording of the output. | (11) |
| | A1 | At least one variable with a suitable variable name | | |
| | A2 | username = bard423 | | |
| | A3 | password = nX2934? | | |
| | A4 | Loop used | | |
| | A5 | Username or password entered | | |
| | A6 | Username or password stored in variable(s) | | |
| | A7 | At least one suitable input message | | |
| | A8 | Checks username and password | | |
| | A9 | Appropriate error message(s) displays | | |
| | A10 | Welcome message displayed | | |
| | A11 | Executing and producing correct output | | |

Code examples

Java

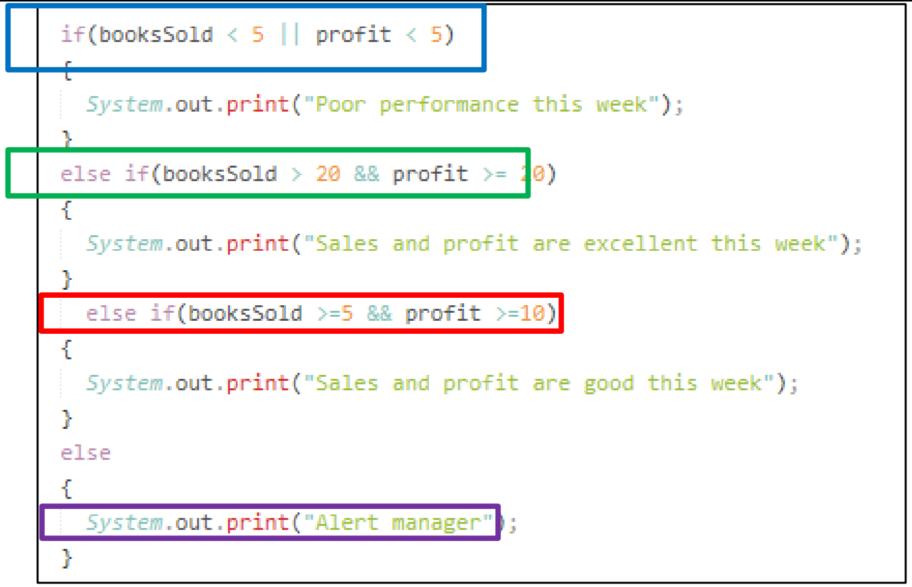


```java
        String username = "bard423";
        String password = "nX2934?";
        int count = 0;
        String inputUsername = "";
        String inputPassword = "";
        Scanner input = new Scanner(System.in);

    // Print prompts, take and check input from user
        while (!inputUsername.equals(username) || !inputPassword.equals(password))
        {
            if (count > 0)
            {
                System.out.println("There is a problem with the login details. Try again");
            }
            count++;
            System.out.print("Enter your username ");
            inputUsername = input.next();
            System.out.print("Enter your password ");
            inputPassword = input.next();
        }
        System.out.println("Welcome");
```

| Question | mp | Answer | | Additional Guidance | Mark |
|---|---|---|---|---|---|
| 2 (b) | Award 1 mark for each correct condition. | | | Alternative alternatives e.g. Line 11 booksSold <=4 etc. | |
| | | **Condition** | **Output message** | | |
| | B1 | Number of books sold is at least 5; profit made is at least 10 | Sales and profit are good this week | | |
| | B2 | Number of books sold is over 20; profit made is at least 20 | Sales and profit are excellent this week | | |
| | B3 | Number of books sold is under 5 or profit made is under 5 | Poor performance this week | | |
| | B4 | All other inputs | Alert manager | | (4) |

Code examples

Java

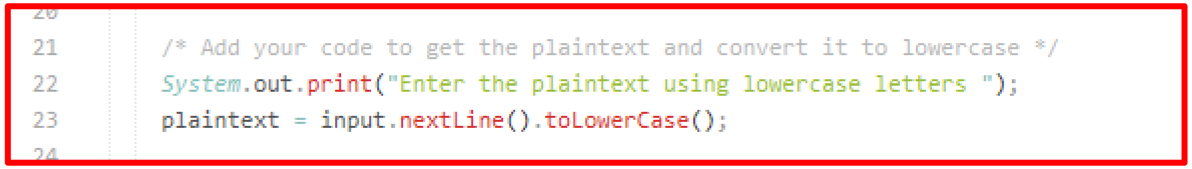


```
if(booksSold < 5 || profit < 5)
{
  System.out.print("Poor performance this week");
}
else if(booksSold > 20 && profit >= 20)
{
  System.out.print("Sales and profit are excellent this week");
}
  else if(booksSold >=5 && profit >=10)
{
  System.out.print("Sales and profit are good this week");
}
else
{
  System.out.print("Alert manager");
}
```

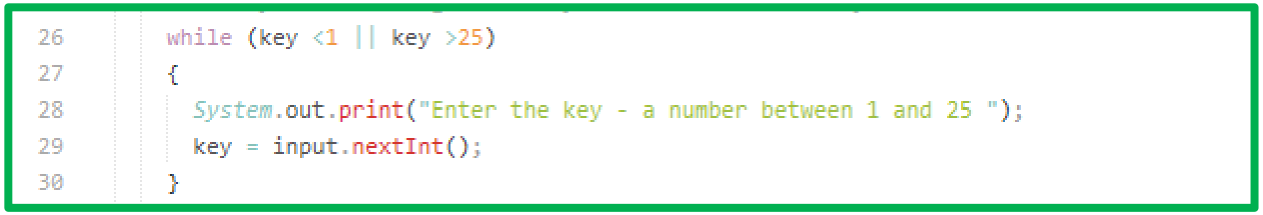
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 3 (a) | A1 | Get plaintext and store in plaintext variable | Accept alternative wording<br>Line numbers may be different compared to the examples shown<br><br>When testing the completed code use lowercase for the input | (1) |
| | A2 | Get key and store in key variable | | (1) |
| | A3 | Validate key | | (1) |
| | A4 | Open file to write | | (1) |
| | A5 | Write cipher text | | (1) |
| | A6 | Close file | | (1) |
| | A7 | Displays ciphertext | | (1) |
| | A7 | Executing and producing correct output to file and screen | | (1) |

Code examples

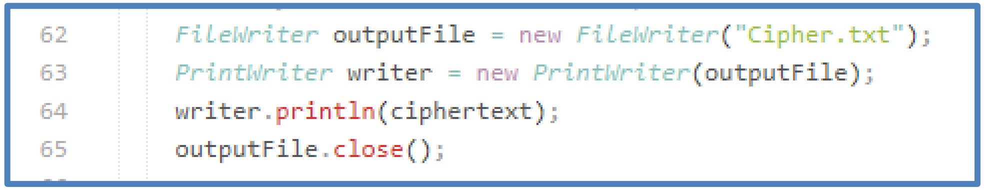
Java



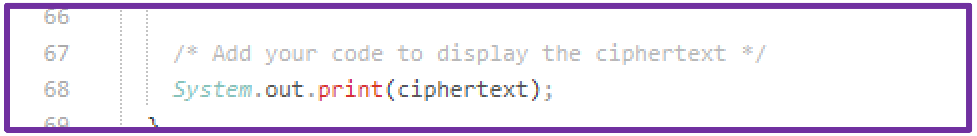
```
/* Add your code to get the plaintext and convert it to lowercase */
System.out.print("Enter the plaintext using lowercase letters ");
plaintext = input.nextLine().toLowerCase();
```



```
while (key <1 || key >25)
{
  System.out.print("Enter the key - a number between 1 and 25 ");
  key = input.nextInt();
}
```



```
FileWriter outputFile = new FileWriter("Cipher.txt");
PrintWriter writer = new PrintWriter(outputFile);
writer.println(ciphertext);
outputFile.close();
```

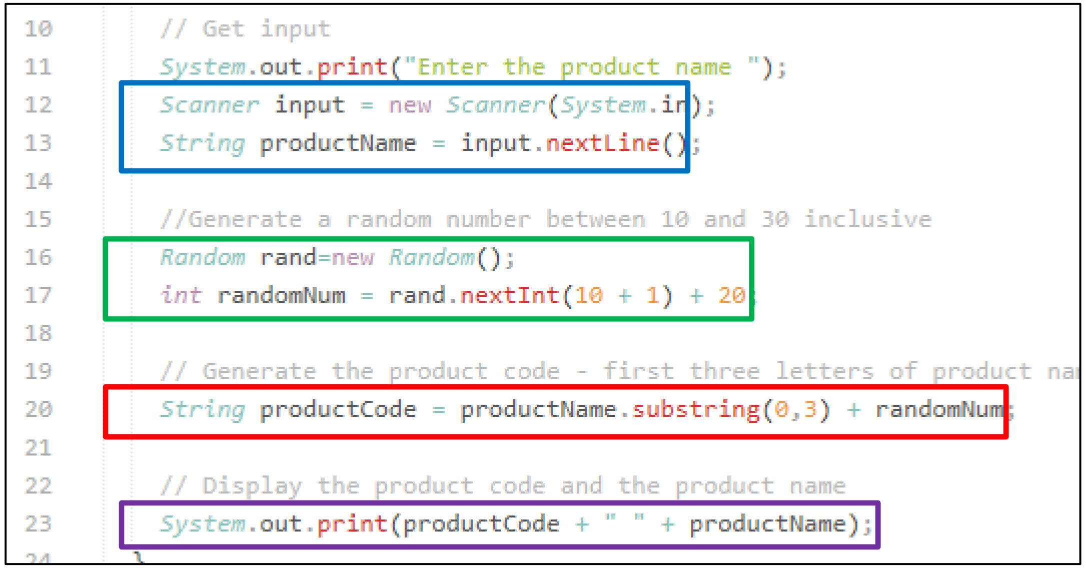


```
/* Add your code to display the ciphertext */
System.out.print(ciphertext);
```

| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| | A1 | At least 1 variable has a meaningful name | Ignore spelling mistakes in input message | (7) |
| | A2 | Product name requested using a suitable input message | | |
| | A3 | Random number generated that would be at least 10 **or** no higher than 30 | | |
| | A4 | Random number generated that would be in the correct range 10 to 30 | | |
| | A5 | First 3 letters of product name generated | | |
| | A6 | First 3 letters of product name and random number concatenated to generate productCode | | |
| | A7 | productCode and productName output in the same print statement | | |

Code examples

Java



```
    // Get input
    System.out.print("Enter the product name ");
    Scanner input = new Scanner(System.in);
    String productName = input.nextLine();

    //Generate a random number between 10 and 30 inclusive
    Random rand=new Random();
    int randomNum = rand.nextInt(10 + 1) + 20;

    // Generate the product code - first three letters of product na
    String productCode = productName.substring(0,3) + randomNum;

    // Display the product code and the product name
    System.out.print(productCode + " " + productName);
  }
```

For Q5, the first 11 marks are for coding that matches requirements of task. The remaining 9 marks should be allocated on a best fit.

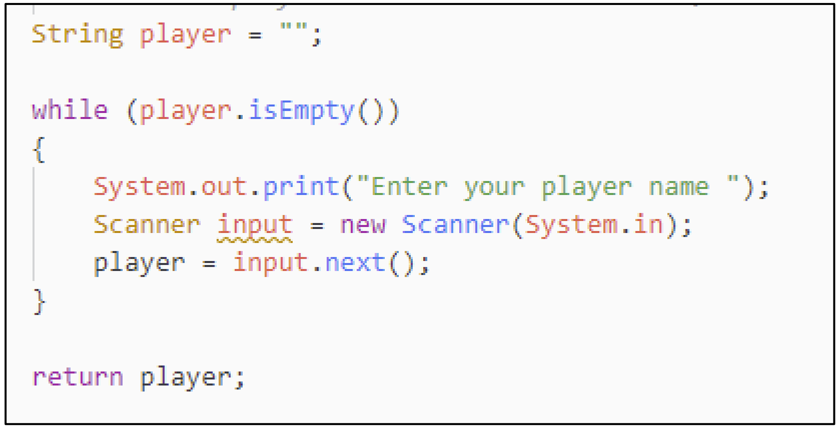
| Question | mp | Answer | Additional Guidance | Mark |
|---|---|---|---|---|
| 5 | **addPlayerName()** | | | |
| | A1 | Suitable prompt for player name and assigned to suitable variable | | |
| | **guessCapital()** | | | |
| | A2 | Ensure question can only be used once | | |
| | A3 | Question includes suitable message and country name | | |
| | A4 | Check made to see if guess is correct | | |
| | A5 | If guess correct score incremented | | |
| | A6 | If guess is incorrect suitable message displayed | | |
| | A7 | If guess incorrect country and its capital concatenated with message | | |
| | A8 | Repeated for five questions | | |
| | **Main Program** | | | |
| | A9 | Player name or score displayed | | |
| | A10 | At least one menuChoice calls correct subprogram | | |
| | A11 | Main program calls all three sub-programs correctly | | (11) |

| Band 1 (1-3 marks) | Band 2 (4-6 marks) | Band 3 (7-9 marks) | Mark |
|---|---|---|---|
| Little attempt to decompose into component parts | Some attempt to decompose into component parts | The problem has been decomposed into component parts | (9) |
| 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 | |

Code examples

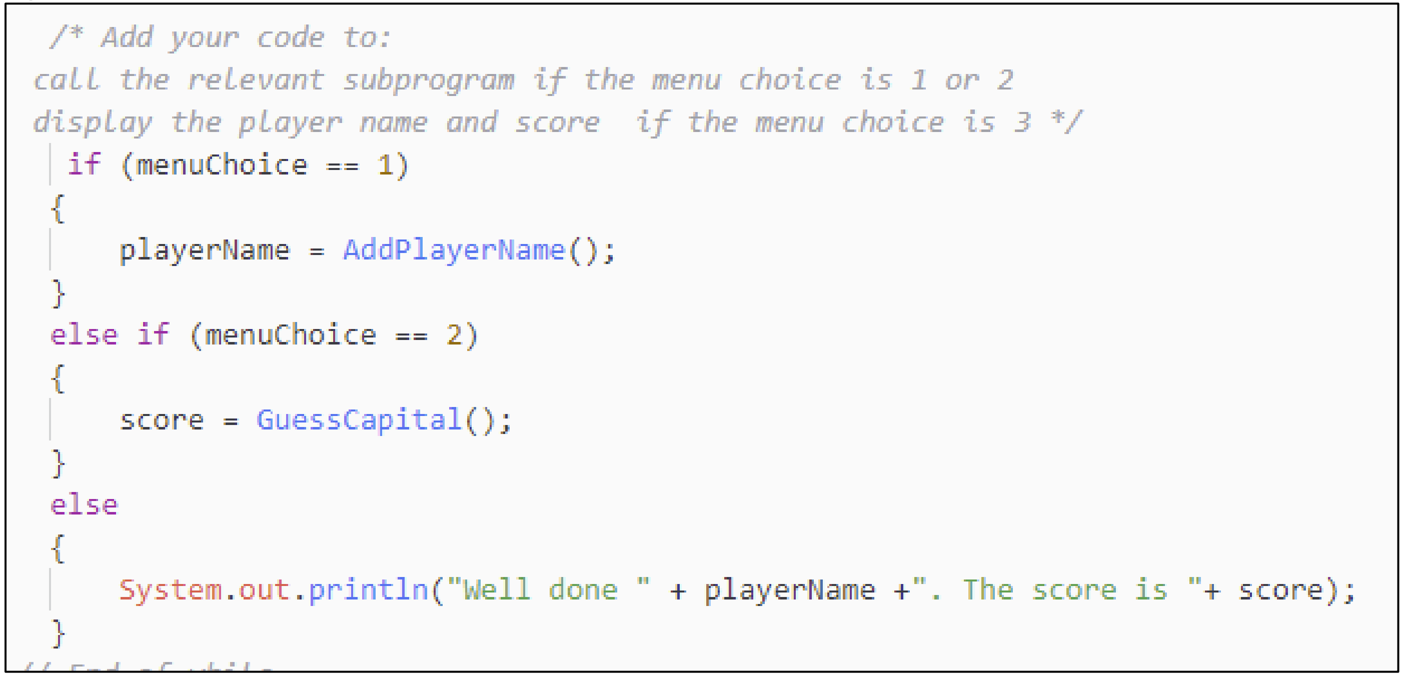
Java

**Add player name function**



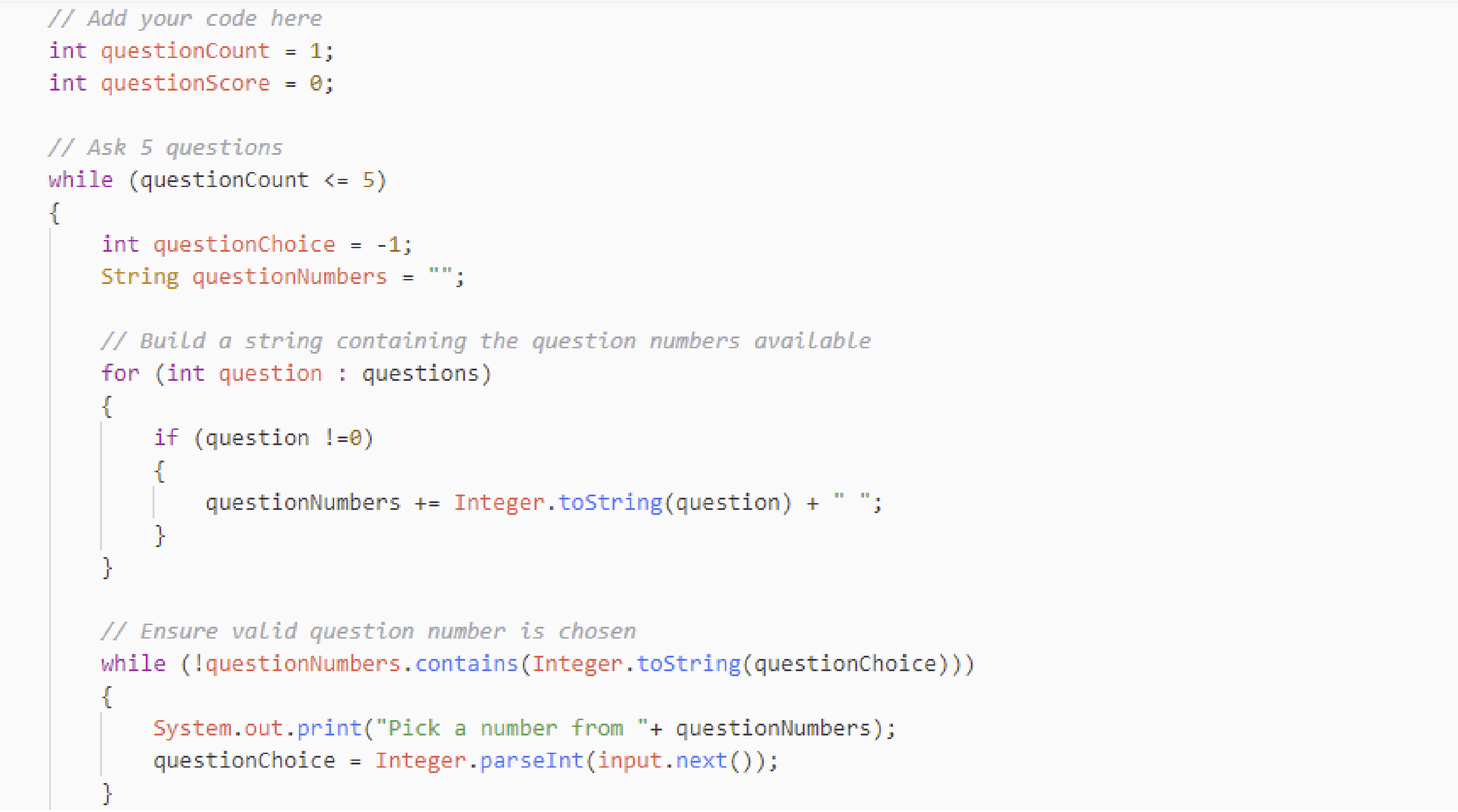
```
String player = "";

while (player.isEmpty())
{
    System.out.print("Enter your player name ");
    Scanner input = new Scanner(System.in);
    player = input.next();
}

return player;
```

**Main program**

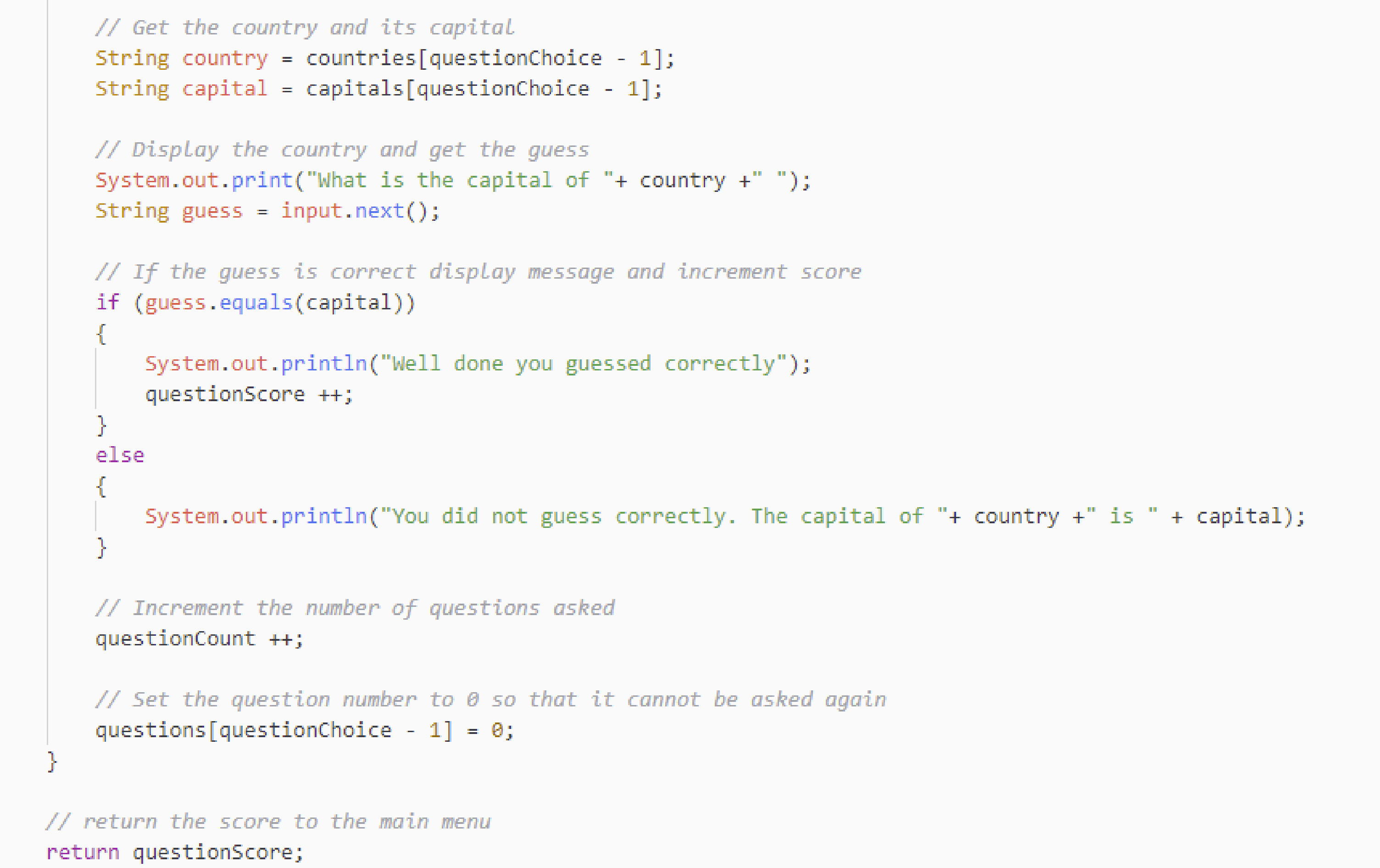


```
 /* Add your code to:
call the relevant subprogram if the menu choice is 1 or 2
display the player name and score  if the menu choice is 3 */
  if (menuChoice == 1)
 {
     playerName = AddPlayerName();
 }
 else if (menuChoice == 2)
 {
     score = GuessCapital();
 }
 else
 {
     System.out.println("Well done " + playerName +". The score is "+ score);
 }
```

**Guess capital city function**



```
// Add your code here
int questionCount = 1;
int questionScore = 0;

// Ask 5 questions
while (questionCount <= 5)
{
    int questionChoice = -1;
    String questionNumbers = "";

    // Build a string containing the question numbers available
    for (int question : questions)
    {
        if (question !=0)
        {
            questionNumbers += Integer.toString(question) + " ";
        }
    }

    // Ensure valid question number is chosen
    while (!questionNumbers.contains(Integer.toString(questionChoice)))
    {
        System.out.print("Pick a number from "+ questionNumbers);
        questionChoice = Integer.parseInt(input.next());
    }
```



```java
    // Get the country and its capital
    String country = countries[questionChoice - 1];
    String capital = capitals[questionChoice - 1];

    // Display the country and get the guess
    System.out.print("What is the capital of "+ country +" ");
    String guess = input.next();

    // If the guess is correct display message and increment score
    if (guess.equals(capital))
    {
        System.out.println("Well done you guessed correctly");
        questionScore ++;
    }
    else
    {
        System.out.println("You did not guess correctly. The capital of "+ country +" is " + capital);
    }

    // Increment the number of questions asked
    questionCount ++;

    // Set the question number to 0 so that it cannot be asked again
    questions[questionChoice - 1] = 0;
}

// return the score to the main menu
return questionScore;
```

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