Section A: Theory Answers

Question 1

1. Method used: ‘Main’
2. String variable: ‘str’
3. Class Name: ‘Test’
4. Namespace used: ‘System’

Question 2

1.Line 14 creates an instance of the **Two** class with **Two two = new Two();**.

* It then calls the **changeValue** method from the base class **One** with the value **10** as an argument and assigns the result to the variable **i**.
* After that, it calls the **addValues** method from the derived class **Two** with arguments **i** and **5** and assigns the result to the variable **ans**.
* Finally, it prints the result using **Console.WriteLine’.**

2. Line 10 declares a new class named **Two** that inherits from the **One** class.

* Inheritance is indicated by the colon (**:**), making **Two** a derived class of **One**.
* This means that **Two** has access to the members (fields, properties, and methods) of the base class **One**.

3.Line 22 defines a method named addValues within the class Two.

* This method takes two integer parameters (val and addVal) and returns their sum (val + addVal).
* The method is used to perform addition, and it is invoked in Line 12 with the values i and 5.
* The result is assigned to the variable ans.

Question 3

a. **#define**: Used for conditional compilation.

b. **Using System**: Allows the use of types from the **System** namespace.

c. **#if…#else…#endif**: Used for conditional inclusion of code.

d. **Main()**: The entry point of the program, executes the specified logic based on conditional compilation.

Question 4

* (a): Initialize **i** before the loop and increment **i** inside the loop.
* (b): Initialize **j** before the inner loop and increment **j** inside the inner loop.
* (c): Condition for the inner loop, specifying when it should execute.
* (d): Initialize **k** before the innermost loop and increment **k** inside the innermost loop.
* (e): Corrected the comparison operator in the **if** statement (**==** instead of just **=**).

Question 5

a. First line of output: "Flower 1: daffodil"

b. Array declaration and initialization: **string[] flowers = { "daffodil", "rose", "tulip" };**

c. Statement used to loop through a certain collection: **foreach (string flower in flowers) { ... }**

d. Statement decrementing int i: **for (int i = flowers.Length - 1; i >= 0; i--) { ... }** e. Class: **class Test**

Question 6

a. Base Class: **Animal**

b. Derived Class: **Dog**

c. New Instance: **Animal a = new Animal();**

d. Entry point of the program: **public static void Main()** in the **Dog** class.

Question 7

1. **Line 1: int age = 20;**
   * This line declares a variable **age** of type **int** and initializes it with the value **20**.
   * It's setting the age variable to 20.
2. **Line 4: if ((age > 15) ^ (gender == 'f'))**
   * This line starts an **if** statement with a condition.
   * The condition **(age > 15) ^ (gender == 'f')** uses the exclusive OR (**^**) operator.
   * It evaluates to **true** if either age is greater than 15 or the gender is 'f', but not both.
   * If the condition is **true**, the code inside the following block (lines 6-7) will be executed.
3. **Line 14: Console.WriteLine("Childish!!");**
   * This line is part of the **else** block and will be executed if the condition in the **if** statement on line 4 is **false**.
   * It prints "Childish!!" to the console.
4. **Line 8: else if ((age > 21) && (gender == 'f'))**
   * This line introduces an **else if** statement with a new condition.
   * The condition **(age > 21) && (gender == 'f')** checks if the age is greater than 21 and the gender is 'f'.
   * If this condition is **true**, the code inside the block (lines 9-11) will be executed.
5. **Line 12: else**
   * This line starts an **else** block, which means that if none of the previous conditions (line 4 or line 8) are **true**, the code inside this block will be executed.
   * In this case, it leads to line 14, printing "Childish!!" to the console.

Question 8

a. **Loop that executes at least once and then executes a group of statements continually while the condition is true.**

* **Answer:** **do-while loop**

b. **Loop used when you know how many times it must be executed.**

* **Answer:** **for loop**

c. **Loop that is designed to execute as long as the condition being checked remains true.**

* **Answer:** **while loop**

d. **Statement used when you want to skip the rest of the current execution and begin the next one immediately.**

* **Answer:** **continue statement**

e. **Used to store a single character.**

* **Answer:** **char** (data type)

Question 9

a. **Evaluates to true if the first value is equal to the second value, false otherwise.**

* **Answer:** **==** (equality operator)

b. **Evaluates to true if the first value is not equal to the second value, false otherwise.**

* **Answer:** **!=** (inequality operator)

c. **Evaluates to true if the first value is greater than or equal to the second value, false otherwise.**

* **Answer:** **>=** (greater than or equal to operator)

d. **Evaluates to true if the first value is smaller but not equal to the second value, false otherwise.**

* **Answer:** **<** (less than operator)

e. **Condition on the left is checked and if it equals to false, then the whole statement equals false without checking the condition on the right.**

* **Answer:** **&&** (logical AND operator)

Question 10

a. **Changes the sign of the value.**

* **Answer:** **-** (Unary Minus)

b. **Decreases the value by one.**

* **Answer:** **--** (Decrement)

c. **Inverts the value of the Boolean variable of the expression.**

* **Answer:** **!** (Logical NOT)

d. **Has no effect.**

* **Answer:** **;** (Semicolon) - In some contexts, a statement terminator that has no effect on the expression.

e. **Subtracts the right operand from the left operand and returns the value.**

* **Answer:** **-** (Subtraction)

**SECTION B**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace C\_\_Assessment

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// Check if TextBox is not empty

if (!string.IsNullOrWhiteSpace(textBoxWord.Text))

{

string newWord = textBoxWord.Text.Trim();

// Check if the word has not been entered more than once

if (!comboBox1.Items.Contains(newWord))

{

// Add the new word to the list

comboBox1.Items.Add(newWord);

// Display a MessageBox informing the user that the new word has been added

MessageBox.Show($"Word '{newWord}' has been added successfully.", "Word Added", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Display the concatenated words in the Label control

DisplayConcatenatedWords();

}

else

{

// Display a MessageBox if the word has been entered more than once

MessageBox.Show("Word already exists in the list.", "Duplicate Word", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

// Display a MessageBox if TextBox is empty

MessageBox.Show("Please enter a word.", "Empty Word", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

// Clear the TextBox for the next input

textBoxWord.Clear();

}

private void DisplayConcatenatedWords()

{

// Concatenate the words in the list

string concatenatedWords = string.Join(" ", comboBox1);

// Display the concatenated words in the Label control

labelConcatenatedWords.Text = $"Concatenated Words: {concatenatedWords}";

}

}

}