USN | | | | | | | | | | |

# Internal Assessment Test 1 – September 2019

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sub: | Dot Net Framework for Application Development | | | | Sub Code: | 17CS564/15CS564 | Branch: | CSE |
| Date: | 7-9-19 | Duration: | 90 min’s | Max Marks: 50 | Sem / Sec: | 5th A,B,C | | OBE |

<u>Answer all FIVE FULL Questions</u>

| | | | MARKS | CO | RBT |
|---|---|---|---|---|---|
| 1 | (a) | List and explain the decision statements used in C#. Write a C# program to print prime numbers between 1 to 1000. | [10] | CO1 | L1 |
| 2 | (a) | How to set optional parameters for a method? Explain with an example. | [05] | CO1 | L2 |
| | (b) | Explain partial classes. Give an example | [05] | CO2 | L3 |
| 3 | (a) | Demonstrate the usage of ‘ref’ and ‘out’ keywords by using them in a program to swap two numbers. | [06] | CO1 | L3 |
| | (b) | Write a C# program to compute the average of N array elements. | [04] | CO1 | L2 |
| 4 | (a) | Define exception. Differentiate between error and exception. List the different classes of exception. Explain try and catch block with example. | [10] | CO2 | L1 |
| 5 | (a) | What is a constructor? Explain with example. List the different types of constructors. | [06] | CO2 | L1 |
| | (b) | What is constructor overloading? Explain with example. | [04] | CO2 | L3 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Sub: | **15CS564/17CS564 - DOTNET FRAMEWORK FOR APPLICATION DEVELOPMENT** | | | | |
| Date: | 09.09.2019 | Duration: | 90mins | Max. Marks | 50 |

**ALL QUESTIONS COMPULSORY. ANSWER ALL QUESTIONS**

|  |  |
|---|---|
| Q1(a) | Listing out decision statements – 2m<br>Explanation – 2m<br>Program – 6m |
| Q2(a) | How to set optional parameters for a method – 2m<br>Example – 3m |
| Q2(b) | Explain partial classes – 2m<br>Example 3m |
| Q3(a) | Program to swap number using ref keyword – 3m<br>Program to swap number using out keyword – 3m |
| Q3(b) | Program to find average:<br>Program outline – 2m<br>Usage of looping statements – 2m |
| Q4(a) | Define exception. – 1m<br>Differentiate between error and exception. – 2m<br>List the different classes of exception. – 4m<br>Explain try and catch block with example. – 3m |
| Q5(a) | What is a constructor? – 1m<br>Explain with example. – 3m<br>List the different types of constructors. – 2m |
| Q5(b) | What is constructor overloading? – 1m<br>Explain with example. – 3m |

| Sub: | Dot Net Framework for Application Development | | Sub Code: | 17CS564/15CS564 | Branch: | CSE |
|---|---|---|---|---|---|---|
| Date: | 9-9-19 | Duration: 90 mins | Max Marks: | 50 | Sem / Sec: | 5th A,B,C |

| | | |
|---|---|---|
| Q1(a) | **List and explain the decision statements used in C#. Write a C# program to print prime numbers between 1 to 1000.** | 2marks for listing each decision making statement<br><br>2 marks for each explanation<br><br>6marks for the program |
| Ans: | The decision making statements used in C# are:<br>1. **if statement**<br>An **if** statement consists of a boolean expression followed by one or more statements.<br>2. **if…else statement**<br>An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is false.<br>3. **nested if statement**<br>You can use one if or else if statement inside another if or else if statement(s).<br>4. **switch statement**<br>A switch statement allows a variable to be tested for equality against a list of values.<br><br>(program below) | |

```
using System;

namespace PrimeNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            bool isPrime = true;
            Console.WriteLine("Prime Numbers : ");
            for (int i = 2; i <= 1000; i++)
            {
                for (int j = 2; j <= i/2; j++)
                {

                    if (i != j && i % j == 0)
                    {
                        isPrime = false;
                        break;
                    }

                }
                if (isPrime)
                {
                    Console.Write("\t" +i);
                }
                isPrime = true;
            }
            Console.ReadKey();
        }
    }
}
```

Q2(a) **How to set optional parameters for a method? Explain with an example.**

Ans:

In C#, a method may contain required or optional parameters. A method that contains optional parameters does not force to pass arguments at calling time. It means we call method without passing the arguments. The optional parameter contains a default value in function definition. If we do not pass optional argument value at calling time, the default value is used.

2 marks for explanation

3 marks for program

EXAMPLE 1:

```
using System;
namespace CSharpFeatures
{
    public class OptionalArgumentsExample
    {
        public static void Main(string[] args)
        {
            add(12,12); // Passing both arguments
            add(10);    // Passing only required argument
        }
        static void add(int a, int b = 10) // second parameter is optional
        {
            Console.WriteLine(a+b);
        }
    }
}
```

EXAMPLE 2:

```
using System;
namespace CSharpFeatures
{
    public class OptionalArgumentsExample
    {
        public static void Main(string[] args)
        {
            add(12,12); // Passing both arguments
            add(12);    // Passing one argument
            add();      // Passing No argument
        }
        static void add(int a = 12, int b = 12) // Making all parameters optional
        {
            Console.WriteLine(a+b);
        }
    }
}
```

Q2(b) **Explain partial classes. Give an example.**

Ans:

A partial class is a special feature of C#. It provides a special ability to implement the functionality of a single class into multiple files and all these files are combined into a single class file when the application is compiled. A partial class is created by using a partial keyword. This keyword is also useful to split the functionality of methods, interfaces, or structure into multiple files.

2marks for explanation

```
P1.cs
public partial class Geeks {
        private string Author_name;
        private int Total_articles;

        public Geeks(string a, int t)
        {
                this.Authour_name = a;
                this.Total_articles = t;
        }
}

P2.cs
public partial class Geeks {
        public void Display()
        {
                Console.WriteLine("Author's name is : " + Author_name);
                Console.WriteLine("Total number articles is : " + Total_articles);
        }
}
```

3marks for program

When we execute the above code, then compiler combines P1.cs and P2.cs into a single file, i.e. P as shown below.

```
public class Geeks {
        private string Author_name;
        private int Total_articles;

        public Geeks(string a, int t)
        {
                this.Authour_name = a;
                this.Total_articles = t;
        }

        public void Display()
        {
                Console.WriteLine("Author's name is : " + Author_name);
                Console.WriteLine("Total number articles is : " + Total_articles);
        }
}
```

Q3(a) **Demonstrate the usage of ‘ref’ and ‘out’ keywords by using them in a program to swap two numbers.**

```
// C# program to illustrate the concept of out parameter
using System;
namespace Demo
{
   class Program
   {
      public static void Main()
      {
         int a, b;
```

3 marks for

out

```
            swap(out a, out b);
            Console.WriteLine("a={0} and b={1}", a, b);
        }

        public static void swap(out int a, out int b)
        {
            a=20;
            b=30;
            Console.WriteLine("a={0} and b={1}", a, b);
            int temp;
            temp = a;
            a=b;
            b=temp;
        }
    }
}
```

3marks for ref

```
// C# program to illustrate the concept of ref parameter
using System;

using System;

namespace CalculatorApplication {
   class NumberManipulator {
      public void swap(ref int x, ref int y) {
         int temp;

         temp = x; /* save the value of x */
         x = y;    /* put y into x */
         y = temp; /* put temp into y */
      }
      static void Main(string[] args) {
         NumberManipulator n = new NumberManipulator();

         /* local variable definition */
         int a = 100;
         int b = 200;

         Console.WriteLine("Before swap, value of a : {0}", a);
         Console.WriteLine("Before swap, value of b : {0}", b);

         /* calling a function to swap the values */
         n.swap(ref a, ref b);

         Console.WriteLine("After swap, value of a : {0}", a);
         Console.WriteLine("After swap, value of b : {0}", b);

         Console.ReadLine();
      }
   }
}
```

| | | |
|---|---|---|
| Q3(b) | **Write a C# program to compute the average of N array elements.** | |

```
using System;
namespace Arrayaverage
{
public class findavg
{
   public static void Main()
   {
            int[] a= new int[10];
            int i, n, sum=0;
            double avg = 0.0;

              Console.Write("Input the number of elements to be stored in the array :");
               n = Convert.ToInt32(Console.ReadLine());

              Console.Write("Input {0} elements in the array :\n",n);
              for(i=0;i<n;i++)
               {
                  Console.Write("element - {0} : ",i);
                         a[i] = Convert.ToInt32(Console.ReadLine());
                 }

            for(i=0; i<n; i++)
            {
               sum += a[i];
            }
            Avg = sum/n;

            Console.Write("Average of all elements stored in the array is : {0}\n\n", avg);
  }
 }
}
```

4marks for program

Q4(a) **Define exception. Differentiate between error and exception. List the different classes of exception. Explain try and catch block with example.**

Ans:

An exception is a problem that arises during the execution of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

1mark for definition

Errors:
- Errors are unexpected issues that may arise during computer program execution.
- Errors cannot be handled.

Exceptions:
- Exceptions are unexpected events that may arise during run-time.
- Exceptions can be handled using try-catch mechanisms.

2 marks for difference

The different classes of exception are: (any four)

1. System.IO.IOException: Handles I/O errors.
2. System.IndexOutOfRangeException: Handles errors generated when a method refers to an array index out of range.
3. System.ArrayTypeMismatchException: Handles errors generated when type is mismatched with the array type.
4. System.NullReferenceException: Handles errors generated from referencing a null object.
5. System.DivideByZeroException: Handles errors generated from dividing a dividend with zero.
6. System.InvalidCastException: Handles errors generated during typecasting.
7. System.OutOfMemoryException: Handles errors generated from insufficient free memory.
8. System.StackOverflowException: Handles errors generated from stack overflow.

4marks for classes

```
using System;
namespace ErrorHandlingApplication {
  class DivNumbers {
    int result;

    DivNumbers() {
      result = 0;
    }
    public void division(int num1, int num2) {
      try {
        result = num1 / num2;
      } catch (DivideByZeroException e) {
        Console.WriteLine("Exception caught: {0}", e);
      } finally {
        Console.WriteLine("Result: {0}", result);
      }
    }
    static void Main(string[] args) {
      DivNumbers d = new DivNumbers();
      d.division(25, 0);
      Console.ReadKey();
    }
  }
}
```

3marks for program

Q5(a) **What is a constructor? Explain with example. List the different types of constructors.**

Ans:

A class constructor is a special member function of a class that is executed whenever we create new objects of that class. A constructor has exactly the same name as that of class and it does not have any return type.

1mark for definition

```
using System;

namespace LineApplication {
  class Line {
    private double length;   // Length of a line

    public Line() {
      Console.WriteLine("Object is being created");
    }
    public void setLength( double len ) {
      length = len;
    }
    public double getLength() {
      return length;
    }

    static void Main(string[] args) {
      Line line = new Line();

      // set line length
      line.setLength(6.0);
      Console.WriteLine("Length of line : {0}", line.getLength());
      Console.ReadKey();
    }
  }
}
```

3 marks for program

Different types of constructors are:

1. Default Constructor
2. Parametrized Constructor

```
public Line() { //Default Constructor
      Console.WriteLine("Object is being created");
    }

public Line(double len) {  //Parameterized constructor
```

2marks for types

```
        Console.WriteLine("Object is being created, length = {0}", len);
        length = len;
    }
```

Q5(b) **What is constructor overloading? Explain with example.**

Ans: When more than one constructor with the same name is defined in the same class, they are called overloaded, if the parameters are different for each constructor.

1 mark for definition

```
using System;
namespace Program {
  class Student {
    private double SubjectOne;
    private double SubjectTwo;
    string StudentName;
    public Student() {
      this.SubjectOne = 80;
    }
    public Student(double SubjectOne) {
      this.SubjectOne = SubjectOne;
    }
    public Student(string StudentName, double SubjectOne, double SubjectTwo) {
      this.SubjectOne = SubjectOne;
      this.SubjectTwo = SubjectTwo;
      this.StudentName = StudentName;
    }
    public double GetSubjectOneMarks() {
      return this.SubjectOne;
    }
    public double GetSubjectTwoMarks() {
      return this.SubjectTwo;
    }
    public string GetStudentName() {
      return this.StudentName;
    }
  }
  class Program {
    static void Main(string[] args) {
      Student s1 = new Student();
      Student s2 = new Student(90);
      Student s3 = new Student("Amit",88, 60);
      Console.WriteLine("One");
      Console.WriteLine("Subject One Marks: {0}", s1.GetSubjectOneMarks());
      Console.WriteLine();
      Console.WriteLine("Second");
      Console.WriteLine("Subject One Marks: {0}", s2.GetSubjectOneMarks());
      Console.WriteLine();
      Console.WriteLine("Third");
      Console.WriteLine("Student name: {0}", s3.GetStudentName());
      Console.WriteLine("Subject One Marks: {0}", s3.GetSubjectOneMarks());
      Console.WriteLine("Subject Two Marks: {0}", s3.GetSubjectTwoMarks());
      Console.ReadKey();
    }
  }
}
```

3 marks for example