Started on	Monday, 4 November 2019, 2:39 PM
State	Finished
Completed on	Monday, 4 November 2019, 3:16 PM
Time taken	37 mins 30 secs
Marks	22.62/26.50
Grade	**17.07** out of 20.00 (**85**%)

Question **1**

Correct

Mark 1.00 out of 1.00

If this code was run, would the comment in the if block be reached?

```
static void Main(string[] args)
{
    bool myBool = false;
    bool yourBool = myBool;
    bool ourBool = ! (myBool || yourBool);

    if( (myBool && ourBool) || yourBool )
    {
        //--Some code--//
    }
}
```

Select one:

- True
- False ✔

The correct answer is 'False'.

Question **2**

Correct

Mark 1.00 out of 1.00

Which is the correct technique for setting the int called myInt to be equal to the first element of the array intArray?

```
static void Main(string[] args)
{
    int myInt;
    int[] intArray = { 2, 3, 4, 5, 6 };
    // -- Insert answer here --
}
```

Select one:

- a. intArray[1] = myInt;
- b. myInt = (int)intArray;
- c. myInt = intArray[0]; ✔
- d. myInt = intArray;
- e. myInt = intArray[1];

Your answer is correct.

In order to access an array, we use the [] operator and place inside of it the index of the element we want to access. Arrays start with index 0, so the first element is intArray[0];

The correct answer is: myInt = intArray[0];

Question **3**

Incorrect

Mark 0.00 out of 1.00

This is an example of implicit casting:

```
int myInt = 5;
double myDouble = (double)myInt;
```

Select one:

- True ✖
- False

The correct answer is 'Fal

enum CardSuit

Question **4**

Incorrect

Mark 0.00 out of 1.00

CardSuit myCardSuit CardSuit

myCardSuit

CardSuit

This is an example of explicit casting below:

```
long myLong = 1000L;
int myInt = (int)myLong;
```

CardSuit Heart

Select one:

- True

CardSuit Club

- False ✖

The correct answer is 'True'.

CardSuit Spade

CardSuit Diamond

default

Question **5**

Correct

Mark 2.00 out of 2.00

enum | myCardSuit | Club | Diamond | Heart | CardSuit | default

Use drag and drop [ ]ogram which uses enums to accomplish the following features:

- Defines an enum called CardSuit, which contains the values {Heart, Club, Spade, Diamond}
- Creates an instance of the CardSuit enum called "myCardSuit"
- Grabs a user int, and casts it to the CardSuit enum type.
- Passes the value to a method OutputCardSuitPrediction, which takes a CardSuit as an argument.
- Outputs a different message depending on the value of the passed argument. Each output message contains the name of the value in it.

```
public [ ] ✔ [ ] ✔ {Heart, Club, Spade, Diamond }
static void Main(string[] args)
{
    [ ] ✔ [ ] ✔ = ([ ] ✔ )int.Parse(Console.ReadLine());
    OutputCardSuitPrediction([ ] ✔ );
}
public static void OutputCardSuitPrediction([ ] ✔ cs)
{
    switch (cs)
    {
        case [ ] ✔ . [ ] ✔ :
            Console.WriteLine("The Heart beats on");
            break;
        case [ ] ✔ . [ ] ✔ :
            Console.WriteLine("We're going CLUBbing tonight");
            break;

        case [ ] ✔ . [ ] ✔ :
            Console.WriteLine("Get the Spade, time to garden.");
            break;
        case [ ] ✔ . [ ] ✔ :
            Console.WriteLine("What is a Diamond worth?");
            break;
```

[ ] ✔ :

```
            Console.WriteLine("Unhandled switch value: " + cs + ", case not handled.");
            break;
    }
}
```

Your answer is correct.

The correct answer is:
Use drag and drop to create a program which uses enums to accomplish the following features:

- Defines an enum called CardSuit, which contains the values {Heart, Club, Spade, Diamond}
- Creates an instance of the CardSuit enum
- Grabs a user int,
- Passes the value to a method OutputCardSuitPrediction, which takes a CardSuit as an argument.
- Outputs a different message depending on the value of the passed argument. Each output message contains the name of the value in it.

```
public [enum] [CardSuit]{Heart, Club, Spade, Diamond }
static void Main(string[] args)
{
    [CardSuit] [myCardSuit] = ([CardSuit])int.Parse(Console.ReadLine());
    OutputCardSuitPrediction([myCardSuit]
}
public static void O
{
    switch (cs)
    {
        case [Ca
            Co
            break;
        case [CardSuit].[Club]:
            Console.WriteLine("We're going CLUBbing tonight");
            break;
```

string int "Amanda" "Mufasa" KeyValuePair string int kv.Key

"Amanda"	Key	"Mufasa"	Hashcode	int	
"Raj"	5	KeyValuePair	Value	kv.Value	
masterDict[kv]					

```
            case [CardSuit].[Spade]:
                Console.WriteLine("Get the Spade, time to garden.");
                break;
            case [CardSuit].[Diamond ]:
                Console.WriteLine("What is a Diamond worth?");
                break;
            [default]:
                Console.WriteLine("Unhandled switch value: " + cs + ", case not handled.");
                break;
        }
}
```

GetHighest params int[]

int

values

highest values[i]

highest

params	string	values	highest	int[]	GetHighest	

Question **6**

Partially correct

Mark 2.70 out of 3.00

Drag and drop to create a program with the following features:

- Contains a dictionary whose key is a string and value is an int.
- Fills the dictionary with the following pairs: <"Amanda", 5>, <"Raj", 18>, <"Mufasa", 22>
- Outputs the value associated with the key "Amanda" to the console
- Removes the value associated with Mufasa
- Then loops through the remaining values and outputs them.

```
public static Dictionary< [ ] , [ ] > masterDict = new Dictionary<
[ ] , [ ] >();
static void Main(string[] args)
{
    masterDict.Add("Amanda", 5);
    masterDict.Add("Raj", 18);
    masterDict.Add("Mufasa", 22);

    Console.WriteLine(masterDict[ ]);
    masterDict.Remove( );
    foreach ( < , > kv in masterDict)
        Console.Write( );
}
```

Your answer is partially correct.

You have correctly selected 9.

The correct answer is:

Drag and drop to create a program with the following features:

- Contains a dictionary whose key is a string and value is an int.
- Fills the dictionary with the following pairs: <"Amanda", 5>, <"Raj", 18>, <"Mufasa", 22>
- Outputs the value associated with the key "Amanda" to the console

- Removes the value associated with Mufasa
- Then loops through the remaining values and outputs them.

```
public static Dictionary<[string], [int]> masterDict = new Dictionary<[string], [int]>();
static void Main(string[] args)
{
        masterDict.Add("Amanda", 5);
        masterDict.Add("Raj", 18);
        masterDict.Add("Mufasa", 22);

        Console.WriteLine(masterDict[ ["Amanda"] ]);
        masterDict.Remove(["Mufasa"]);
        foreach ([KeyValuePair]<[string], [int]> kv in masterDict)
                Console.Write([kv.Value]);
}
```

Question **7**

Correct

Mark 2.00 out of 2.00

Create a method called "GetHighest" which given ANY number of ints, returns the highest one.

```
static void Main(string[] args)
{
      Console.Write("Highest: " + GetHighest(3, 4, 5, 6, 7, 23, 3));
}
public static int [ ✔ ] ( [ ✔ ] [ ✔ ] values)
{
      [ ✔ ] highest = int.MinValue;
      for(int i =0; i < [ ✔ ] .Length; i++)
      {
            [ ✔ ] = (highest < values[i]) ? [ ✔ ] : highest;
      }
      return [ ✔ ] ;
}
```

Your answer is correct.

The correct answer is:
Create a method called "GetHighest" which given ANY number of ints, returns the highest one.

```
static void Main(string[] args)
{
      Console.Write("Highest: " + GetHighest(3, 4, 5, 6, 7, 23, 3));
}
public static int [GetHighest]([params] [int[]] values)
{
      [int] highest = int.MinValue;
      for(int i =0; i < [values].Length; i++)
      {
            [highest] = (highest < values[i]) ? [values[i]] : highest;
      }
      return [highest];
}
```

Question **8**

Partially correct

Mark 1.00 out of 2.00

For each letter (A, B, C, ...), select whether the corresponding code would or would not be executed.

Pay attention to the use of the **virtual**, **override**, and **new** keywords in the method signatures. Also pay attention to the data types of the objects created in Main(), as well as to the the data types of the reference variables used to access those objects.

```
static void Main()
{
    Animal animal = new Animal();
    Animal tiger = (Animal) new Tiger();

    animal.Eat();
    tiger.Walk();
    tiger.Sleep();
    tiger.Drink();
}

class Animal
{
    public virtual void Walk()
    {
        // A
    }

    public virtual void Eat()
    {
        // B
    }

    public virtual void Drink()
    {
        // C
    }

    public virtual void Sleep()
    {
        // D
    }
}

class Tiger : Animal
{
    public override void Walk()
    {
        // E
    }

    public override void Eat()
    {
        // F
    }

    public override void Drink()
    {
        base.Drink();
        // G
    }

    public new void Sleep()
    {
```

```
        // H
    }
}
```

D	Code is not executed.	✗
B	Code is not executed.	✗
G	Code is executed.	✓
F	Code is executed.	✗
H	Code is not executed.	✓
A	Code is not executed.	✓
E	Code is executed.	✓
C	Code is not executed.	✗

Your answer is partially correct.

You have correctly selected 4.
The correct answer is: D → Code is executed., B → Code is executed., G → Code is executed., F → Code is not executed., H → Code is not executed., A → Code is not executed., E → Code is executed., C → Code is executed.

Question **9**

Partially correct

Mark 2.75 out of 3.00

Consider the following code:

```
static void Main(string[] args)
{
     int a = 10;
     double b = 20;
     float c = 3.5f;
     string s = "huehue";

     //--Method call--//
}

//--Method definition--//
```

For the pairs below, state whether the syntax is legal (whether it will compile and run without error).

In each case, the first statement is the method definition, the second is the method call.

Example:

The following option ...

static void MyMethod(int a) { }
MyMethod()

... would mean that the result would look like...

```
static void Main(string[] args)
{
     int a = 10;
     double b = 20;
     float c = 3.5f;
     string s = "huehue";

     MyMethod()   // Method call
}

static void MyMethod(int a) { }   // Method definition
```

This would be Invalid, because it contains a compiler error.

static void MyMethod(string b) { } MyMethod(a.ToString());	Valid ✔
static void MyMethod(int b, double a) { } MyMethod((int)b, (double)a);	Invalid ✖
static void MyMethod() { } MyMethod(a, b);	Invalid ✔
static MyMethod(void) { } MyMethod();	Invalid ✔
static void MyMethod(int a, int b) { } MyMethod(5, a);	Valid ✔
static int MyMethod(int a, double b) { return (int)(a+b); } MyMethod(MyMethod(5, b), b);	Valid ✔
static void MyMethod(int b, double a) { } MyMethod(a, b);	Valid ✔
static void MyMethod(int b = 7) { } MyMethod(-7);	Valid ✔
static void MyMethod(int b, double a, string s) { } MyMethod(a, b);	Invalid ✔
static int MyMethod(int a, double b) { return (int)(a+b); } MyMethod(int a, double b);	Invalid ✔
static void MyMethod(int a, int b = 7) { } MyMethod(a);	Valid ✔
static void MyMethod(int b = 7) { } MyMethod();	Valid ✔

Your answer is partially correct.

You have correctly selected 11.
The correct answer is: static void MyMethod(string b) { }
MyMethod(a.ToString()); → Valid, static void MyMethod(int b, double a) { }
MyMethod((int)b, (double)a); → Valid, static void MyMethod() { }
MyMethod(a, b); → Invalid, static MyMethod(void) { }
MyMethod(); → Invalid, static void MyMethod(int a, int b) { }
MyMethod(5, a); → Valid, static int MyMethod(int a, double b) { return (int)(a+b); }
MyMethod(MyMethod(5, b), b); → Valid, static void MyMethod(int b, double a) { }
MyMethod(a, b); → Valid, static void MyMethod(int b = 7) { }
MyMethod(-7); → Valid, static void MyMethod(int b, double a, string s) { }
MyMethod(a, b); → Invalid, static int MyMethod(int a, double b) { return (int)(a+b); }
MyMethod(int a, double b); → Invalid, static void MyMethod(int a, int b = 7) { }

MyMethod(a); → Valid, static void MyMethod(int b = 7) { }
MyMethod(); → Valid

Question **10**

Correct

Mark 1.50 out of 1.50

- Assume a class called Employee that has a single constructor which has the following parameters in order: string name, double hourlySalary
- We want to make a child class called Manager whose constructor has 3 parameters: string name, double hourlySalary, int departmentNumber
- We want the Manager constructor to properly pass the two arguments, name and hourlySalary, to the parent constructor.
- We want to make a class called BigBoss that is a child of Manager. We want BigBoss's constructor to only take name as a parameter, and then pass the name along with the values 100 and 0 to the parent (Manager) constructor.

Which of the options below are valid for the constructors of Employee, Manager, and BigBoss?

There should be a total of 3 correct answers. (One constructor for each class.)

Select one or more:

- [ ] a. public Manager(string name, double hourlySalary, int departmentNumber) { //..... }
- [ ] b. public BigBoss(string name, double hourlySalary, int deptNumber) : base(name, hourlySalary, deptNumber) { //..... }
- [ ] c. public Manager(string name, double hourlySalary, int departmentNumber) : Employee(name,hourlySalary) { //..... }
- [x] d. public BigBoss(string name) : base(name, 100, 0) { //..... } ✔
- [ ] e. public Manager(int departmentNumber) : Employee() { //..... }
- [ ] f. public Manager(string name, double hourlySalary, int departmentNumber) : parent(name,hourlySalary) { //..... }
- [ ] g. public Manager(string name, double hourlySalary, int departmentNumber) : Employee() { //..... }
- [ ] h. public Manager(int departmentNumber) : Employee(name,hourlySalary) { //..... }
- [ ] i. public Manager(string name, double hourlySalary, int departmentNumber) : Employee() { //..... }
- [ ] j. public Employee(string name, double hourlySalary) : base(name, hourlySalary) { //..... }
- [x] k. public Manager(string name, double hourlySalary, int departmentNumber) : base(name,hourlySalary) { //..... } ✔
- [ ] l. public BigBoss(string name) : Employee(name, 100, 0) { //..... }
- [ ] m. public Employee(public string name, public double hourlySalary){ //..... }

- [x] n. public Employee(string name, double hourlySalary){ //..... } ✔
- [ ] o. public BigBoss(string name, double hourlySalary, int deptNumber) : parent(name, hourlySalary, deptNumber) { //..... }
- [ ] p. public Manager(int departmentNumber) : Employee(string name, int hourlySalary) { //..... }
- [ ] q. public BigBoss(public string name, public double hourlySalary = 100, public int deptNumber = 0) : parent(name, hourlySalary, deptNumber) { //..... }

Your answer is correct.

The correct answers are: public Manager(string name, double hourlySalary, int departmentNumber) : base(name,hourlySalary) { //..... }, public BigBoss(string name) : base(name, 100, 0) { //..... }, public Employee(string name, double hourlySalary){ //..... }

Question **11**

Correct

Mark 1.00 out of 1.00

```
public delegate int MyDelegate(int a, int b);

static void Main()
{
      MyDelegate myDelegate = Sum(5, 10);
      Console.Write(myDelegate.Invoke(50, 10));
}

public static int Sum(int a, int b) { return a + b;  }
```

If there is no error, what will be displayed in the console?

If there is an error, what caused the error?

Select the only true answer.

Select one:

- a. Error. The correct way to set the delegate would be: MyDelg myDelg += Sum(5,10);
- b. Error. Delegates do not exist in C#.
- c. No errors. The output will be 15.
- d. Error. Sum(5, 10) causes the method to be called. The proper syntax for assigning the method reference to the delegate would be: MyDelegate myDelegate = Sum; ✔
- e. Error. There is no instance of the delegate type created. The type was only declared.
- f. No errors. The output will be 60.
- g. Error. The myDelegate.Invoke(...) call does not need arguments, since the arguments were defined when the delegate object was was initialized.
- h. Error. The delegate signature is invalid. Delegates can only point to methods with void return types.

Your answer is correct.

The correct answer is: Error. Sum(5, 10) causes the method to be called. The proper syntax for assigning the method reference to the delegate would be:
MyDelegate myDelegate = Sum;

Question **12**

Correct

Mark 1.00 out of 1.00

True or false: After this code is run, there will be one MyStruct object with three MyStruct variables referencing it.

```
struct MyStruct {}

static void Main (string[] args)
{
    MyStruct ms1 = new MyStruct();
    MyStruct ms2 = ms1;
    MyStruct ms3 = ms2;
}
```

Select one:

- True
- False ✔

The correct answer is 'False'.

Question **13**

Correct

Mark 1.00 out of 1.00

What number will be written to the console by the last line of code in the Main() method?

```
static void Main()
{
    int[] myArr   = new int[2] { 10, 20 };
    int totalSum = 0;

    Swap(myArr);
    totalSum += myArr[0];

    myArr  = new int[2] { 10, 20 };    //resets the array in case it changed
    Swap(myArr[0],myArr[1]);
    totalSum += myArr[0];

    myArr   = new int[2] { 10, 20 };   //resets the array in case it changed
    Swap2(myArr);                          //calls Swap2 function, not Swap
    totalSum += myArr[0];

    Console.WriteLine(totalSum);
}

public static void Swap(int[] array)
{
    int t = array[0];
    array[0] = array[1];
    array[1] = t;
}

public static void Swap(int a, int b)
{
    int t = a;
    a = b;
    b = t;
}


public static void Swap2(int[] array)
{
    int a = array[0];
```

```
    int b = array[1];
    Swap(a, b);
}
```

Answer: 40 ✔

The first Swap method:
public static void Swap(int[] array)
It is given a reference type variable. Therefore, changes made to elements of the array through the array parameter inside the method change the array object itself (i.e., the elements of the array).

Overload of the Swap method:
public static void Swap(int a, int b)
Its parameters' types are value types, so the values of the arguments passed in are simply copied into the parameters. Therefore, changes made to a and b inside the method do not change the values of the variables that were passed as arguments.

Swap2 method
public static void Swap2(int[] array)
Swap2 is given an array, which is a reference type. However, it copies the array's contents into two value-type variables, a and b, and then those variables are copied again into the parameters of the Swap method.

The correct answer is: 40

Question **14**

Partially correct

Mark 1.67 out of 2.00

Select all the statements about interfaces that are true.

Select one or more:

- [x] a. Classes can implement any number of interfaces. ✔
- [x] b. It is possible for any interface type to be used as the key type of a dictionary. ✔
- [x] c. A reference to an object, whose class implements some interface called ISomeInterface, can be cast into the type ISomeInterface. ✔
- [ ] d. Interfaces can be instantiated with the new keyword.
- [ ] e. Interfaces are like ints, but with faces.
- [x] f. Interfaces can inherit from other interfaces. ✔
- [x] g. An instance method declared inside an interface cannot contain a body. ✔
- [ ] h. A concrete class that implements some interface must give a definition for each abstract method in the interface.
- [ ] i. Interfaces can contain instance method definitions and variable fields.

Your answer is partially correct.

You have correctly selected 5.
The correct answers are: Classes can implement any number of interfaces., Interfaces can inherit from other interfaces., A reference to an object, whose class implements some interface called ISomeInterface, can be cast into the type ISomeInterface. , An instance method declared inside an interface cannot contain a body., It is possible for any interface type to be used as the key type of a dictionary., A concrete class that implements some interface must give a definition for each abstract method in the interface.

Question **15**

Correct

Mark 1.00 out of 1.00

Given this method signature:

public void Method(int a = 0, int b = 0)

Select all of the valid method calls from the following:

Select one or more:

- [x] a. Method(); ✔
- [x] b. Method(2); ✔
- [ ] c. Method(2, 3, 4, 5);
- [x] d. Method(2, 3); ✔
- [ ] e. Method(2, 3, 4, 5, 6);
- [ ] f. Method(2, 3, 4);

Your answer is correct.

The correct answers are: Method();, Method(2);, Method(2, 3);

Question **16**

Correct

Mark 1.00 out of 1.00

Given this method signature:

public void Method(int a, int b, params int[] c)

Select all of the valid method calls from the following:

Select one or more:

- [ ] a. Method();
- [x] b. Method(2, 3, 4, 5); ✔
- [x] c. Method(2, 3, 4); ✔
- [x] d. Method(2, 3); ✔
- [ ] e. Method(2);

Your answer is correct.

The correct answers are: Method(2, 3);, Method(2, 3, 4);, Method(2, 3, 4, 5);

Question **17**

Correct

Mark 1.00 out of 1.00

Consider the following two classes.

```
public class ClassA
{
   public ClassA(int a, string b)
   {
      //...
   }
}

public class ClassB : ClassA
{
   // ClassB constructor
}
```

Select which of the following would be the correct syntax for a ClassB constructor that calls the parent class constructor:

Select one:

a.
```
public ClassB(int a, string b)
{
   base(a, b);
}
```

b.
```
public ClassB(int a, string b)
{
   super(a, b);
}
```

c.
```
public ClassB(int a, string b) : super(a, b)
{
}
```

d.
```
public ClassB(int a, string b)
{
   ClassA(a, b);
}
```

e.
```
public ClassB(int a, string b) : ClassA(a, b)
{
}
```

f.
```
public ClassB(int a, string b) : base(a, b)
{
}
```
✔

Your answer is correct.

The correct answer is:
```
public ClassB(int a, string b) : base(a, b)
{
}
```

Question **18**

Correct

Mark 1.00 out of 1.00

Consider the following code.

```
public struct Point
{
   public double x;
   public double y;

   public Point(double x, double y)
   {
      this.x = x;
      this.y = y;
   }
}
```

After executing the following code, what **x** and **y** values will **pointB** contain?

```
public static void Main()
{
   Point pointA = new Point(0, 0);
   Point pointB = pointA;
   pointB.x = 10;
   pointA.x = 5;
   pointA.y = 5;
}
```

Select one:

- a. x = 0, y = 0
- b. x = 10, y = 10
- c. x = 10, y = 0 ✔
- d. x = 0, y = 5
- e. x = 5, y = 5
- f. x = 10, y = 5

Your answer is correct.

The correct answer is: x = 10, y = 0

◀ Interface exercise

Jump to...

Announcements ▶