Dashboard / My courses / D05-C#-Web1901B-BChevalier / General / Final Theoretical Examination

Started on	Friday, 23 August 2019, 3:14 PM
State	Finished
Completed on	Friday, 23 August 2019, 4:16 PM
Time taken	1 hour 2 mins
Marks	22.67/25.00
Grade	**18.13** out of 20.00 (**91**%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

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'.

string int

Question **2**

Correct

Mark 2.00 out of 2.00

Drag and drop to create a program with the following fea "Bob"

- Contains a dictionary whose key is a string "Moofasa" int.
- Fills the dictionary with the following pairs: <"Bob",5>, <"Sandy",18>, <"Moofasa",22>
- Outputs the value associated w KeyValuePair " to the string int
- Removes the value associated with Moofa kv.Value
- Then loops through the remaining values and outputs them.

string | int | kv.Value | "Moofasa" | "Bob" | k

public static Dictionary< ✓ , ✓ > masterDict = new Dictionary<

✓ Value | Hashcode

static void Main(string[] args)

{

masterDict.Add("Bob", 5);

masterDict.Add("Sandy", 18);

masterDict.Add("Moofasa", 22);

Console.WriteLine(masterDict[✓]);

masterDict.Remove(✓);

foreach (✓ < ✓ , ✓ > kv in masterDict)

Console.Write(✓);

}

Your answer is correct.

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: <"Bob",5>, <"Sandy",18>, <"Moofasa",22>
- Outputs the value associated with the key "Bob" to the console
- Removes the value associated with Moofasa

- 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("Bob", 5);
        masterDict.Add("Sandy", 18);
        masterDict.Add("Mocfasa", 22);
                                    enum              CardSuit

        Console.WriteLine(masterDict[ ["Bob"] ]);
        masterDict.Remove(["Moofasa"]);
        foreach ([KeyValueI  CardSuit  [int]>  myCardSuit )ict)        CardSuit
                Console.Write([kv.Value]);
                                                       myCardSuit
}
```

CardSuit

CardSuit Heart

CardSuit Club

CardSuit Spade

CardSuit Diamond

default

enum	CardSuit	myCardSuit	Club	Heart	Spade

Question **3**

Correct

Mark 2.00 out of 2.00

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 called "myCardSuit"
- Grabs a user int, and casts it into an the CardSuit enum.
- Passes the value to a function, which takes a CardSuit as an argument.
- Outputs a different message depending on the value of the passed arguement. 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;
    }
}
```

GetHighest params int

int

highest _values[i]

Your answer is correct. highest

The correct answer is:

Use drag and drop _values[i] highest GetHighest params int eatures:

- 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 into an the CardSuit enum.
- Passes the value to a function, which takes a CardSuit as an argument.
- Outputs a different message depending on the value of the passed arguement. 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 OutputCardSuitPrediction([CardSuit] cs)
{
        switch (cs)
        {
                case [CardSuit].[Heart]:
                        Console.WriteLine("The Heart beats on");
                        break;

                case [CardSuit].[Club]:
                        Console.WriteLine("We're going CLUBbing tonight");
                        break;

                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;
        }
    }
```

Question **4**

Correct

Mark 2.00 out of 2.00

Create a function 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 < _values.Length; i++)
      {
            ________ ✓ = (highest < _values[i]) ?________ ✓ :highest;
      }
      return ________ ✓ ;
}
```

Your answer is correct.

The correct answer is:
Create a function 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 **5**

Correct

Mark 2.00 out of 2.00

Match each piece of code to the statement that applies to it.

```
class SomeClass
{
    public void SomeFunc() { SomeFunc(); }
}
```

This code will cause a run-time exception. ✓

```
class SomeClass
{
    public void SomeFunc() {}
    public new void SomeFunc() { int myInt =
5; }
}
```

This is an example of a compiler error. ✓

```
class SomeClass
{
    public void SomeFunc() {}
}
class OtherClass : SomeClass
{
    public new void SomeFunc() {}
}
```

This is an example of function hiding. ✓

```
class SomeClass
{
    public void SomeFunc() {}
    public void SomeFunc(float someFloat) {}
}
```

This is an example of function overloading ✓

Your answer is correct.

The correct answer is: class SomeClass
{
    public void SomeFunc() { SomeFunc(); }
} → This code will cause a run-time exception., class SomeClass
{

```
    public void SomeFunc() {}
    public new void SomeFunc() { int myInt = 5; }
} → This is an example of a compiler error., class SomeClass
{
    public void SomeFunc() {}
}
class OtherClass : SomeClass
{
    public new void SomeFunc() {}
} → This is an example of function hiding., class SomeClass
{
    public void SomeFunc() {}
    public void SomeFunc(float someFloat) {}
} → This is an example of function overloading
```

Question **6**

Correct

Mark 2.00 out of 2.00

Match each piece of code to the statement that applies to it.

public Person Somebody { get; } ✓	This is an example of a readonly property.
private House MyHouse { get { return value; } } ✓	This is an example of a compiler error.
protected int SomeInt { get; private set; } ✓	This property can only be read by this class and classes that inherit from this class.
private float someFloat; public float SomeFloat { get { return SomeFloat; } set { someFloat = value; } } ✓	This property will cause a run-time error when it is read.

Your answer is correct.

The correct answer is: public Person Somebody { get; } → This is an example of a readonly property., private House MyHouse { get { return value; } } → This is an example of a compiler error., protected int SomeInt { get; private set; } → This property can only be read by this class and classes that inherit from this class., private float someFloat;
public float SomeFloat { get { return SomeFloat; } set { someFloat = value; } } → This property will cause a run-time error when it is read.

Question **7**

Correct

Mark 2.00 out of 2.00

- 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)

Select one or more:

☐ a. public BigBoss(string name, double hourlySalary, int deptNumber) : base(name, hourlySalary, deptNumber) { //..... }

☑ b. public Manager(string name, double hourlySalary, int departmentNumber) : base(name,hourlySalary) { //..... } ✓

☐ c. public Manager(int departmentNumber) : Employee(string name, int hourlySalary) { //..... }

☐ d. public BigBoss(string name, double hourlySalary, int deptNumber) : parent(name, hourlySalary, deptNumber) { //..... }

☐ e. public BigBoss(string name) : Employee(name, 100, 0) { //..... }

☐ f. public BigBoss(public string name, public double hourlySalary = 100, public int deptNumber = 0) : parent(name, hourlySalary, deptNumber) { //..... }

☑ g. public BigBoss(string name) : base(name, 100, 0) { //..... } ✓

☐ h. public Employee(string name, double hourlySalary) : base(name, hourlySalary) { //..... }

☐ i. public Manager(string name, double hourlySalary, int departmentNumber) : Employee(name,hourlySalary) { //..... }

☐ j. public Manager(string name, double hourlySalary, int departmentNumber) : Employee() { //..... }

☐ k. public Employee(public string name, public double hourlySalary){ //..... }

☐ l. public Manager(string name, double hourlySalary, int departmentNumber) { //..... }

☑ m. public Employee(string name, double hourlySalary){ //..... } ✓

☐ n. public Manager(string name, double hourlySalary, int departmentNumber) : Employee() { //..... }

- o. public Manager(string name, double hourlySalary, int departmentNumber) : parent(name,hourlySalary) { //..... }
- p. public Manager(int departmentNumber) : Employee(name,hourlySalary) { //..... }
- q. public Manager(int departmentNumber) : Employee() { //..... }

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 **8**

Correct

Mark 1.00 out of 1.00

```
public delegate int MyDelg(int a, int b);
static void Main(string[] args)
{
        MyDelg myDelg = Sum(5,10);
        Console.Write(myDelg.Invoke(50, 10));
}
public static int Sum(int a, int b) { return a + b;  }
```

Select the only true answer

Select one:

- a. No errors, the output will be 60
- b. Error, the proper way to set the delegate would be MyDelg myDelg += Sum(5,10);
- c. The delegate signature does not match, delegates can only point to functions with void return types
- [✓ d. Error, the Sum(5, 10) causes an invoke to occur. The proper syntax to assign the function reference to the delegate is: MyDelg myDelg = Sum;] ✓
- e. Error, the invoke does not need arguments since the arguments were defined when the function pointer was initialized
- f. No errors, the output will be 15
- g. Error, there is no instance of the delegate created, it was only defined.
- h. Delegates are not real and do not exist, you cannot use delegates in c#

Your answer is correct.

The correct answer is: Error, the Sum(5, 10) causes an invoke to occur. The proper syntax to assign the function reference to the delegate is: MyDelg myDelg = Sum;

Question **9**

Partially correct

Mark 1.67 out of 2.00

Select all the statements about interfaces that are true.

Select one or more:

- ☐ a. It is possible for any interface type to be used as the key type of a dictionary.
- ☑ b. Classes can implement any number of interfaces. ✓
- ☐ c. Interfaces are instantiated with the new keyword.
- ☐ d. Interfaces can contain functions and variables.
- ☑ e. A function declared inside an interface cannot contain a body. ✓
- ☑ f. A class that implements some interface must include each function that exists in the interface. ✓
- ☑ g. Interfaces can inherit from other interfaces. ✓
- ☑ h. A reference to an object of a class that implements some interface called ISomeInterface can be cast into the type ISomeInterface. ✓
- ☐ i. Interfaces are like ints, but with faces.

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 of a class that implements some interface called ISomeInterface can be cast into the type ISomeInterface. , A function 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 class that implements some interface must include each function that exists in the interface.

Question **10**

Correct

Mark 2.00 out of 2.00

Below we have the following code

```
static void Main(string[] args)
{
     Shape[] shapes = new Shape[3];
     shapes[0] = new Square(2);
     shapes[1] = new Square(4);
     shapes[2] = new Rectangle(2,3);
     int totalArea = 0;
     for(int i = 0; i < 3; i++)
     {
         totalArea += shapes[i].GetArea();
     }
    Console.WriteLine("totalArea: " + totalArea);
}
abstract class Shape
{
    private virtual int GetArea() { return 0; }
}

class Square : Rectangle
{
    public Square(int _x) : base(_x,_x) {}
}


class Rectangle : Shape
{
    protected int x, y;
    public Rectangle(int _x, int _y) : Shape()
    {
        x = _x;
        y = _y;
    }
    public override int GetArea() { return x * y; }
}
```

Select all the following options which are TRUE...

Select one or more:

- ☐ a. The program will compile properly and run without error, the output of the program will be: totalArea: 26
- ☐ b. Shape is an abstract class. Therefore we cannot have an array of Shapes, since we cannot create instances of Shape.
- ☐ c. This is a compiler error because the Square constructor passes the same variable twice to the base Shape constructor, which requires no arguments.
- ☐ d. This is a compiler error because the array can only store instances of Shape, but we are trying to put instances of the children classes Rectangle and Square into it.
- ☑ e. Virtual functions cannot be private, since they are made to overridden by children. ✓
- ☐ f. There will be an index out of range exception caused by the for loop
- ☐ g. This is a runtime error because the Square class does not contain an override for the virtual function GetArea(), so when it is called during the for loop, it will crash with an uninitialized function error.
- ☐ h. This is an error because arrays can only be filled using object initializer syntax
- ☐ i. This is a compiler error because the Square constructor passes the same variable twice to the base Rectangle constructor, which requires two unique arguments.
- ☑ j. Rectangle calls Shape constructor incorrectly. Compiler error. ✓

Your answer is correct.

The correct answers are: Rectangle calls Shape constructor incorrectly. Compiler error., Virtual functions cannot be private, since they are made to overridden by children.

Question **11**

Correct

Mark 1.00 out of 1.00

What number will be output to the console by this code?

```
public delegate void MyDelg(out int output, int input = 5);

class Program
{
    static void Main(string[] args)
    {
        int result = 3;
        MyDelg myDelg = new MyDelg(MyFunc);
        myDelg.Invoke(out result);
        Console.WriteLine(result);
    }

    static void MyFunc(out int output, int input)
    {
        output = input;
    }
}
```

Answer: 5 ✓

The correct answer is: 5

Question **12**

Correct

Mark 1.00 out of 1.00

Assuming a class called animal exists, how many elements does this list contain by the end of the code?

```
static void Main(string[] args)
{
    List<Animal> animals = new List<Animal>() { new Animal(), new Animal() };
    Animal toAdd = new Animal();
    Animal sameAnimal = toAdd;
    animals.Add(toAdd);
    animals.Add(sameAnimal);
    animals.Remove(sameAnimal);
    animals.Remove(sameAnimal);
    animals.Remove(sameAnimal);
}
```

Answer: 2 ✓

Here two animals are added in the initializer, and then two identical instances of an animal are added to the list. Lists can contain multiple copies of the same element, there is no unique check, just like arrays. We then ask to remove the instance that matches sameAnimal, so that instance is remove. We then ask to remove the instance that matches some animal, the one that we added "toAdd" IS the same instance as sameAnimal, since they both point to the same instance, so it removes that one as well. Then we attempt to remove it again, however no instance is found, since we already removed both of them, so there remain the first two elements that were added in the initializer.

The correct answer is: 2

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 function?

```
static void Main(string[] args)
{
    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[] _myArr)
{
    int t = _myArr[0];
    _myArr[0] = _myArr[1];
    _myArr[1] = t;
}

public static void Swap(int v1, int v2)
{
    int t = v1;
    v1= v2;
    v2 = t;
}



public static void Swap2(int[] _myArr)
{
    int v1 = _myArr[0];
    int v2 = _myArr[1];
    Swap(v1,v2);
}
```

Answer: 40 ✓

The first Swap function:
public static void Swap(int[] _myArr)
It is given a reference type variable. Therefore, changes made to myArr inside the function change the object itself.

Overload of the Swap function:
public static void Swap(int v1, int v2)
Its parameters are value type, so the values of the arguments passed in are simply copied into the parameters. Therefore, changes made to v1 and v2 inside the function do not change the values of the variables that were passed as arguments.

Swap2 function
public static void Swap2(int[] _myArr)
Swap2 is given an array, which is a reference type variable. However, it copies the array's contents into two value variables, v1 and v2, and then those variables are copied again into the parameters of the Swap function.

The correct answer is: 40

Question **14**

Correct

Mark 1.00 out of 1.00

What number will this code output to the console?

```
static void Main(string[] args)
{
    int int1 = 0;
    int int2 = 0;

    while(false || false)
    {
        int1++;
    }

    do
    {
        int2--;
    }
    while(false && true);

    Console.WriteLine(int1 + int2);
}
```

Answer: -1 ✓

The correct answer is: -1

Question **15**

Correct

Mark 1.00 out of 1.00

Will the exception thrown by this code be handled?

```
class MyCustomException : Exception {}
class MyOtherCustomException : MyCustomException {}

class Program
{
    static void Main(string[] args)
    {
        try { MyFunc(); }
        catch(MyCustomException e) {}
    }

    static void MyFunc()
    {
        MyOtherFunc();
    }

    static void MyOtherFunc()
    {
        throw new MyOtherCustomException();
    }
}
```

Select one:

[✓ True] ✓

False

The correct answer is 'True'.

Question **16**

Incorrect

Mark 0.00 out of 1.00

Will this function correctly return a validated user int without any errors?

```
int GetUserInt()
{
    bool successfulParse = false;
    do
    {
        int toReturn = 0;
        successfulParse = int.TryParse( Console.ReadLine(), out toReturn );
    }
    while(successfulParse);
    return toReturn;
}
```

Select one:

[✓ True] ✗

False

The int toReturn is declared in the wrong scope (it should be outside the do block), so the line **return toReturn;** will throw an error. Also, the while condition should be (!successfulParse).

The correct answer is 'False'.

Question **17**

Correct

Mark 1.00 out of 1.00

Will the if block be entered (in other words, will the Console.WriteLine be run)?

```
static void Main(string[] args)
{
    bool a = true;
    bool b = false;
    bool c = a || b;
    if(((a && c && (a || b)) || b) && !b)
        Console.WriteLine("True");
```

Select one:

[✓ True] ✓

False

The correct answer is 'True'.

◀ Final Project

Jump to...

Final Practical Examination ▶