Practice Final (with solutions)

2024-09-19

Contents

The final exam will be a closed-book paper exam without a calculator. Exam questions will be similar in type to those found here, but fewer in number. While this practice exam is a good study guide, we highly recommend being familiar with *all the material* (including but not limited to your previous exams, labs, projects, quizzes and homework) as well.

Problem 0 (Warm-up)

1. What is the escape sequence for a new line?
   Solution
   `\n`
2. What type is the result of `8 * 12M`?
   Solution
   `decimal`
3. What is the return type of a constructor?
   Solution

There isn't one.

4. What operator would you use to see if int a and int b are equal?
   Solution
   ==
5. List 4 datatypes.
   Solution
   `string`, `int`, `byte`, `decimal`, `double`, `float`, `char`, `bool`, `long`, any user-defined type (class), etc.
6. List 4 reserved words (keywords).
   Solution
   `new`, **`static`**, **`if`**, **`else`**, **`switch`**, **`break`**, any datatype (other than user-defined), etc. Anything that was dark green on any of the slides.
7. What is the difference between a variable and a constant?
   Solution
   variables can have their values changed while constants are set exactly once.
8. Write a statement that declares a constant of type int named `DaysInWeek` and sets its value to 7.
   Solution
   ```
   const int DaysInWeek =7;
   ```
9. In an exam class, if I want to keep track of the total number of exams should the attribute be static or non-static?
   Solution
   static
10. What operator is used to find out the remainder from division?
    Solution
    modulo (%)
11. Write a condition that evaluates to true if an int length is between 4 and 16, both inclusive.
    Solution
    ```
    (length>=4 && length<=16)
    ```
12. How many times would a for loop with this header run? `for(int i=5;i<12; i++)`
    Solution
    7 times.
13. Write a statement or statements that creates an int array of size 50 with each index containing that index as its value. (i.e. 0 at `[0]`, 13 at `[13]`, 49 at `[49]`, etc.).
    Solution
    ```
    int[] numbers = new int[50];
    for(int i= 0; i<numbers.Length;i++)
    {
      numbers[i]=i;
    }
    ```
14. Write a statement or statements to create a random number generator called `examRand` and use it to generate a random number

between 40 and 57 (inclusive).
Solution
```
Random examRand = new Random();
examRand.Next(40,58);
```

Problem 1

Consider the code below:

```
class VirtualPet{
    private string name = "Blank";          // Name of the
     ↪ pet.
    private decimal hungerLevel = 1m;       // Level of
     ↪ hunger, with 1 being full, in percent.
    private decimal happinessLevel = 1m;    // Level of
     ↪ happiness, in percent

    public void SetName(string nameP)
    {
        name = nameP;
    }
}
```

1. Write a statement to instantiate a `VirtualPet` object called `firstPet`.
   Solution
   `VirtualPet firstPet = new VirtualPet();`
   Review classes and objects if you cannot do this. It should be straightforward.
2. Write a getter for the name attribute.
   Solution
   Review classes and objects if you cannot do this. It should be straightforward.
3. Write a statement that would display to the screen the name of the `firstPet` object you created previously. What would be displayed?
   Solution
   Make sure you call the `GetName` method. It should return the default name from our `VirtualPet` class (what is that?).
4. Write a setter for the `hungerLevel` attribute that takes one decimal. The argument should be assigned to the `hungerLevel` attribute only if it is between 0 and 1 (both included), otherwise the attribute should get the value 0.
   Solution
   ```
   public void SetHunger(decimal level)
   ```

```
    {
      hungerLevel=(level>=0m && level<=1m)?level:0m;
    }
```

Note that while we use the conditional operator here, you can replace that with an **if-else**.

5. Draw the UML diagram for the `VirtualPet` class, including the methods you just added.
   Solution

```
|=====================================================|
|                   **VirtualPet**                    |
|-----------------------------------------------------
--|
| - name : string                                     |
| - hungerLevel : decimal                             |
| - happinessLevel : decimal                          |
|-----------------------------------------------------
--|
| + SetName(nameP : string) :                         |
| + GetName( ) : string                               |
| + SetHunger(level : decimal)                        |

|=====================================================|
```

6. Write a constructor that takes 3 arguments (`string`, `decimal`, `decimal`) for the `VirtualPet` class. Your constructor should be such that if one of the decimal arguments is not between 0 and 1 (both included), then 0 gets assigned to both decimal attributes.
   Solution

```
  public VirtualPet(string nameP, decimal hunger,
   ↪  decimal happy)
  {
    name = nameP;
    if(hunger>=0m && hunger<=1m && happy>=0m &&
     ↪  happy<=1m){
      hungerLevel=hunger;
      happinessLevel=happy;
    }
    else{
      hungerLevel=0m;
      happinessLevel=0m;
    }
  }
```

7. Your earlier statement that created the firstPet object will no longer compile after you add the constructor. Why is this the case?
   Solution

Because the default constructor was replaced with the new constructor. Since you are providing your own constructor, C# doesn't provide the default, no-args constructor anymore.

8. Write a statement that would create a new `VirtualPet` object called `secondPet` using the constructor you just added (the argument values are up to you).
   Solution
   "` VirtualPet secondPet = new VirtualPet("Rover", 0.8m, 0.5m);
9. Write a `ToString` method for the `VirtualPet` class. It should display the name, `hungerLevel`, and `happinessLevel`. (Bonus) Display `hungerLevel` and `happinessLevel` graphically: for instance, if `hungerLevel` is at 4.5, display "Hunger: XXXX". You may freely use symbols as if they were normal letters.
   Solution
   ```
   public override string ToString(){
     string returnable= "Name: "+name+ ", Hunger: ";
     for(int i=10; i>0; i--){
       returnable+=(i>(hungerLevel10))? "" : "X";
     }
     returnable+= ", Happiness: ";
     for(int j=10; j>0; j--){
       returnable+=(j>(happinessLevel10))? "" : "X";
     }
     return returnable;
   }
   ```
   Note that while we use the conditional operator here, you can replace that with an **`if-else`**.
10. Write a statement that would use the `ToString` method from the `VirtualPet` class you just added to display information about the `secondPet` object.
    Solution
    `Console.WriteLine(secondPet);`
    This statement will *implicitely* calls the `ToString` method. It is actually equivalent to `Console.WriteLine(secondPet.ToString());`.

Problem 2

This question will have you partially design, implement and use class to represent hamburgers. A Burger has a name, a price, a Boolean for dairy, and a type (typically beef, pork, chicken, veggie).

1. Draw the UML diagram for the Burger class, assuming it contains the listed attributes, a getter for the name attribute and a setter for the price attribute. Do not include any other methods.
   Solution

Assume name is string, price is decimal, and type is string. Otherwise look at the UML from question 1 for an example.

2. Write a getter for the name attribute.
   Solution
   Review classes and objects if you cannot do this. It should be straightforward.
3. Write a setter for the price attribute.
   Solution
   Review classes and objects if you cannot do this. It should be straightforward.
4. Write a constructor that takes 4 arguments and sets the value of the attributes to be the value of the arguments.
   Solution
   ```
   public Burger(string nameP, decimal priceP, bool dairyP; string typeP)     {
   ```
5. Write an additional constructor that takes a name, a dairy, and a type. The price should then be set according to the following table. If the value for type is not in the table, price should be set to -99.99.
   Solution
   ```
   public Burger(string nameP, bool dairyP; string typeP)   {     name=nameP;
   ```
6. Write a static method Promotion that takes as an argument a price and returns a value 75% of the argument.
   Solution
   ```
   public static decimal Promotion (decimal value)
   {
     return(value0.75m);
   }
   ```
7. Write a `ToString` method. The string returned should contain the values of all attributes.
   Solution
   Easier version of `ToString` from Problem 1. Remember to use keyword override.
8. Write a statement/statements that:

- Displays the result of passing 12.84 to Promotion.
- Instantiates a Burger object named OldBeefy with the values "Old Beefy", 1.99, true, and "beef".
- Changes the price of OldBeefy to 2.29.
- Displays the name (and only the name) of OldBeefy.
- Store the value returned by calling the ToString method with OldBeefy in a variable.
  Solution
  ```
  // Displays the result of passing 12.84 to Promotion.
  Console.WriteLine(Burger.Promotion (12.84m));
  // The answer is 9.63m
  ```

```
// Instantiates a Burger object named OldBeefy with
 ↪  the values "Old Beefy", 1.99, true, and "beef".
Burger OldBeefy = new Burger("Old Beefy", 1.99m,
 ↪  true, "beef");

// Changes the price of OldBeefy to 2.29.
OldBeefy.SetPrice(2.29m);

// Displays the name (and only the name) of OldBeefy.
Console.WriteLine(OldBeefy.GetName());

// Store the value returned by calling the ToString
 ↪  method with OldBeefy in a variable.
string holder = OldBeefy.ToString();
```

Problem 3

Complete the table based on the code.

x	y	z	Displays
-1	'e'	18.2M	
-1	'a'	-2	
0	'c'	4.6M	
1	'd'	2	
-1	'b'	115	
1	'd'	-33.7M	
0	'a'	0	
1	'c'	13	
			5

```
int x;
char y;
decimal z;

// x, y, and z are given legal values

if(x<0 && y == 'a'){
  Console.Write("1");
}
else if(z%2==0){
  Console.Write("2");
}
else if(y=='c' || y=='d'){
  Console.Write("3");
```

```
}
else if(x!=0 && z!=0){
  Console.Write("4");
}
else{
  Console.Write("5");
}
```

Solution

x	y	z	Displays
-1	'e'	18.2M	4
-1	'a'	-2	1
0	'c'	4.6M	3
1	'd'	2	2
-1	'b'	115	4
1	'd'	-33.7M	3
0	'a'	0	2
1	'c'	13	3
0	'b'	1	5

Any set of inputs that produce 5 are fine for the last row. This should include 0 for x, anything other than 'a', 'c', or 'd' for y, and anything odd or with a decimal portion for z.

Problem 4

Given two int arrays of equal length, write a code segment that compares the values at each index to see if they match. Return the total number of matches.

Solution

```
//given int [] A and int [] B of some length
int matches=0;
for (int i=0; i<A.Length; i++)
{
  matches+=(A[i]==B[i])?1:0;
}
Console.WriteLine(matches);

//Note that while I use the conditional operator here,
 ↪  you can replace that with an if-else

//if version:
```

```
int matches=0;
for (int i=0; i<A.Length; i++)
{
  if (A[i]==B[i])
  matches++;
}
Console.WriteLine(matches);
```

Problem 5 (Deceptively hard)

Given two string arrays (array A and array B) of unknown (possibly different) lengths, determine if there are any values found in both A and B. If they exist, display them to the screen. At the end of the program, display the total number of common values between A and B. If there are repeating values in either or both arrays, each should only be counted once.

Solution

```
string[] C = new string[A.Length];
string temp="";
bool inC=false, inD=false;
int firstBlankC=0, firstBlankD=0, total=0;

for(int i=0;i<A.Length;i++){
  inC=false;
  for(int j=0;j<C.Length;j++){
    if(A[i]==C[j]){
      inC=true;
      break;//ends the inner for loop early
    }
    if(!inC){//same depth as the inner for loop
    {
      C[firstBlankC]=A[i];
      firstBlankC++;
  }
}//close outer for

//Repeat that code, but replace A with B and C with D.
 ↪  That gets rid
of the duplicates.

for(int i=0;i<firstBlankC;i++){
  for(int j=0;j<firstBlankD;j++){
    if(C[i]==D[j]){
```

```
      Console.WriteLine(C[i]);
      total++;
    }
  }
}
Console.Writeline($"Total values in common: {total}.");
```

(Bonus): How could Lists be used to make this problem easier?

Solution

```
//Assuming A and B are lists instead of arrays; you can
 ↪  also just make
new Lists from the arrays
//with the .AddRange() method of the List class

int total=0;
while(A.Count>0){
if(B.Contains(A[0])){
Console.WriteLine(A[0]);
total++;
}
B.RemoveAll(item => item==A[0]);
A.RemoveAll(item => item==A[0]);
}
Console.WriteLine($"The total number of matches is
 ↪  {total}");
```

Problem 6

Write a program that declares an int variable called "pin" and asks the user for their pin. As long as the user enters something that is not a number, is negative, or greater than 9999, your program should ask again.

(Bonus): Your code should make sure that the pin has exactly 4 digits, including leading zeros.

Solution

```
string userInput = "";
int pin = 0, numDigits = 0;
bool valid = false;
do {
  Console.WriteLine("Please enter your 4-digit pin.");
  userInput = Console.ReadLine();
  valid = int.TryParse(userInput, pin);
  if (valid) {
    valid = (userInput.Length == 4);
```

```
  }
} while (!valid || pin < 0 || pin > 9999);
Console.WriteLine("Pin successfully set!");
```

Problem 7

1. Write a statement that would create an int array of size 100.
   Solution
   ```
   int myArray = new int[100];
   ```
2. Write a series of statements that would ask the user to enter a value for each cell in the array (no need to perform user-input validation, but you may if you like).
   Solution
   ```
   for(int i =0; i<myArray.Length; i++)
   {
     Console.WriteLine($"Enter value {i}.");
     myArray[i]=int.Parse(Console.ReadLine());
   }
   ```
3. Write a series of statements that would ask the user to enter a value, displaying "In your array" if the value is in your array.
   Solution
   ```
   Console.WriteLine("Enter a value to check against
   ↪  your array.");
   int userValue=int.Parse(Console.ReadLine());
   bool inArray=false;
   for (int i =0; i<myArray.Length;i++){
     if(myArray[i]==userValue){
       inArray=true;
     }
   }
   if(inArray){
     Console.WriteLine("In your array");
   }
   ```
4. Write a series of statements that would display the sum of values in the array.
   Solution
   ```
   int sum=0;
   for (int i =0; i<myArray.Length;i++){
   sum+=myArray[i];
   }
   Console.WriteLine($"Sum of array values is {sum}");
   ```
5. Write a series of statements that would display the product of all the non-zero values in the array.
   Solution

```
int product=1;
for(int i =0; i<myArray.Length; i++){
  if(myArray[i]!=0)
  {
    product=myArray[i]
  }
}
Console.WriteLine($"Product of non-zero values is
↪  {product}");
```

6. Write a series of statements that would display the smallest index of the greatest value in the array.
   Solution

```
int greatest=0;
gIndex=0;
for(int i =0; i<myArray.Length;i++)
{
  if(myArray[i]>greatest){
    greatest=myArray[i];
    gIndex=i;
  }
}
Console.WriteLine($"The smallest index of the
↪  greatest value is  {gIndex}");
```