

CSCI 1301 – Exam 1 - Debriefing

You can download the exams for Section A¹ and for Section B². Note that they were different, but of equivalent difficulty. Even if the two exams were different, it is possible to make the following comments regardless of the “version” of the exam you took.

1 Problem 1

For this first problem, you needed to remember that

- **a declaration** is the act of creating and naming a variable, a statement of the form `int myAge;`
- **an assignment** is the act of setting the value of a variable, a statement of the form `myAge = 12;`
- **an instantiation** is the act of creating an object from a class (and assigning its value to a variable, actually, in our examples), a statement of the form `Rectangle myRec = new Rectangle();`
- **an initialization** is a declaration and an assignment combined in one statement, a statement of the form `int myAge = 12;`

I consider those to be extremely basic notions. Failing at this problem is unfortunately an extremely worisome sign...

2 Problem 2

This was about implicit and explicit conversion. Not an easy topic, but you had plenty of opportunities to practice and learn.

3 Problem 3

This exercise, about displaying various strings, had some tricky aspects. Note that, for instance, `Console.WriteLine("myVar");`, would simply display “myVar” followed by a new line, and not the value possibly held by this variable. All of the pitfalls could be detected with a carefull reading of the statements – something I like to encourage you to do.

4 Problem 4

Those exercises were extremely close to what you had in the homework. It was a way of making sure that you had practised those.

5 Problem 5

This one was difficult, as it asked you to write a small program from start to finish. Having successfully completed the project was the best way to get prepared for it.

¹[exam_1_A.pdf](#)

²[exam_1_B.pdf](#)

5.1 Section A

The statement reads:

Complete the following code (you only have to add statements in the body of the Main method), so that your program would 1. Ask the user how many glasses of lemonade they want, 2. Ask the user how many cookies they want, 3. Display (nicely!) on the screen the total price, knowing that glasses of lemonade are \$1 each, cookies are \$1.50 each, and that the sales taxes are 10% of the total.

A possible solution was:

```
decimal total = 0; // Variable used to hold the total due.
Console.WriteLine ("How many glasses of lemonade would you like?");
total += decimal.Parse(Console.ReadLine()); // Lemonade is $1 a glass.
Console.WriteLine ("How many cookies would you like?");
total += decimal.Parse(Console.ReadLine()) * 1.5M; // Cookies are $1.5 each.
total *= 1.10M; // We add the taxes.
Console.WriteLine($"Your total is {total:C}."); // We display the total
↳ nicely, using the format specifier.
```

5.2 Section B

The statement reads:

Write a program that asks the user their name, their age (in years), and displays at the screen the name they entered, followed by their age in days. You should consider that a year is 365.25 days, to account for leap years, but display only a whole number.

A possible solution was:

```
string name; // Variable to hold the user's name.
int ageInYears; // Variable to hold the user's age in years.
Console.WriteLine ("What is your name?");
name = Console.ReadLine(); // We read from the user their name.
Console.WriteLine ("What is your age?");
ageInYears = int.Parse(Console.ReadLine()); // We read and convert from the
↳ user their age in years.
Console.WriteLine($"{name}, you are {(int)(ageInYears * 365.25)} days
↳ old!"); // We compute, truncate, and display the age in days.
```

6 Problem 6

Both problems asked to develop fairly straightforward classes. The answers can be read from the following code:

```

// Section A
class Item {
    private decimal price; // To hold the price of the item.
    private string description; // To hold a description of the item.
    public decimal GetPrice() {
        return price;
    }
    public void SetDescription(string descriptionP) {
        description = descriptionP;
    }
    public decimal GetMultiplePrice(int quantity) {
        return quantity * price;
    }
    // The following was not asked
    public void SetPrice(decimal priceP) {
        price = priceP;
    }
}

// Section B
class Bottle {
    private string content; // To hold a description of the content.
    private int oz; // To hold the capacity in ounces.
    public int GetOz() {
        return oz;
    }
    public void SetContent(string contentP) {
        content = contentP;
    }
    public double GetPint() {
        return oz / 16;
    }
    // The following was not asked
    public void SetOz(int ozP) {
        oz = ozP;
    }
}

using System;

class Welcome {
    static void Main() {
        // Section A
        Item myItem = new Item();
        // The following was not asked
        myItem.SetPrice(19.99M);
        Console.WriteLine($ "{myItem.GetPrice():C}");
        Console.WriteLine("Please, enter a description.");
        myItem.SetDescription(Console.ReadLine());
        Console.WriteLine($ "{myItem.GetMultiplePrice(5):C}");
    }
}

```

```
// Section B
Bottle myBottle = new Bottle();
// The following was not asked
myBottle.SetOz(32);
Console.WriteLine($ "{myBottle.GetOz()}");
Console.WriteLine("Please, enter a content description.");
myBottle.SetContent(Console.ReadLine());
Console.WriteLine($ "{myBottle.GetPint()}");
}
}
```

As an exercise, try to set up those three pieces of codes (that is, the Item, Bottle, and Welcome classes) in a project in Visual Studio, and check that it works as expected.