

# CSCI 1301 – Exam 2 - Debriefing

You can download the exams for Section A<sup>1</sup> and for Section B<sup>2</sup>. Note that they were different, but of equivalent difficulty.

Below, find possible solutions and general comments.

## 1 Problem 1

### 1.1 Section A

```
// The next 4 lines were not needed in your answer.
char iniLName;
decimal pricePaid;
iniLName = 'K'; // Dummy value, change it to test.
pricePaid = 420M; // Dummy value, change it to test.

if(pricePaid > 600M){Console.WriteLine("You are in first class");}
else if (iniLName < 'H'){Console.WriteLine("You are in second class,
    ↪ left-side aisle.");}
else {Console.WriteLine("You are in second class, right-side aisle.");}
```

You may (rightfully) think “Hey, but I didn’t know characters have an order”, and would be right. But you could also work out a solution using switch or if, like:

```
if(pricePaid > 600M){Console.WriteLine("You are in first class");}
else if (iniLName == 'A' || iniLName == 'B' ||
    iniLName == 'C' || iniLName == 'D' ||
    iniLName == 'E' || iniLName == 'F' ||
    iniLName == 'G' || iniLName == 'H'){
    Console.WriteLine("You are in second class, left-side aisle.");
}
else {Console.WriteLine("You are in second class, right-side aisle.");}
```

### 1.2 Section B

Using format specifiers and getting the conditions right were more important than getting the loop correct for that exercise.

```
int age;
do{
    Console.WriteLine("Please, enter your age.");
}while(! int.TryParse(Console.ReadLine(), out age) || age < 0);

if (age > 5 && age < 19){Console.WriteLine($"It's {5:C}.");}
else if (age > 18 && age < 64){Console.WriteLine($"It's {12:C}.");}
else{Console.WriteLine("It's free for you.");}
```

---

<sup>1</sup>exam\_2\_A.pdf

<sup>2</sup>exam\_2\_B.pdf

## 2 Problem 2

You can use the code below to test your answers:

```
// The three values below are dummy.
// Change them to test your answers.
string citizenship = "US";
decimal income = 200M;
int age = 20;

switch (citizenship) {
    case("US"):
    case ("CA"):
        if (income > 100)
            if (age < 21) Console.WriteLine("Go to office A.");
            else if (age < 60) Console.WriteLine("Go to office B.");
            else Console.WriteLine("Go to office C.");
        else
            Console.WriteLine("Go to office D.");
        break;
    case ("DE"):
        if (income > 200 && age > 18) Console.WriteLine("Go to office E.");
        else Console.WriteLine("Go to office F.");
        break;
    case ("FR"):
        if (age <= 18 || income <= 10) Console.WriteLine("Go to office G.");
        else if (income > 200) Console.WriteLine("Go to office H.");
        break;
    default:
        if (age > 21) Console.WriteLine("Go to office I.");
        else Console.WriteLine("Go to office J.");
        break;
}
```

## 3 Problem 3

The two solutions are fairly similar. Short code was privileged below, also because they show an original use of `do while` loops: the multiplication actually takes place *before* asking for a value, so that the multiplication will be performed only if the user actually gave a value to multiply. Of course, for the first run of the loop, we multiply by the “dummy” value 1.

### 3.1 Section A

```
int product = 1;
int answer = 1;
do{
    product *= answer;
```

```

    Console.WriteLine("Enter a number, or 0 (zero) to quit.");
    answer = int.Parse(Console.ReadLine());
}while(answer != 0);
Console.WriteLine("The product of the numbers you entered is " + product +
    ↵ ".");

```

### 3.1.1 Section B

```

int product = 1;
int answer = 1;
bool isNumber;
do{
    product *= answer;
    Console.WriteLine("Enter a number, or anything else to quit.");
    isNumber = int.TryParse(Console.ReadLine(), out answer);
}while(isNumber);
Console.WriteLine("The product of the numbers you entered is " + product +
    ↵ ".");

```

## 4 Problem 4

For both sections, refer to the solution given in lab 12<sup>3</sup>. The version given to Section B was actually a simplification of that exercise.

## 5 Problem 5

The questions were identical for both sections:

- A constructor does *not* have a return type.
- The signature of a method is its name and the ordered list of its input type. So, the signature of the given method is `Scale, int`.
- Method overloading is when two methods have the same name (but different signatures, of course).
- `public Square() {}` is a constructor for that class that takes no argument.
- The `ToString()` method is implicitly called when we try to display an object using `Console.WriteLine`.
- The UML diagram would like like this:

Square
- dimension : int
+ GetDimension() : int
+ SetDimension(dimensionP : int) : void

<sup>3</sup><https://spots.augusta.edu/caubert/teaching/2020/fall/csci1301/weekly/12/lab/#solution>