This page is intentionally left blank. Please, after you indicated your name above, tear off the last page and read it.

____ / **45 pts.** **Problem 1** *Complete Problem 1 (displayed on the last page) below.*

____ / 35 pts. **Problem 2** Suppose given two 1-dimensional arrays of `char` called `passcode` and `codeentered` of the same size.

1. Write a program that displays "Passcode correct" if the content of `passcode` and of `codeentered` are identical.

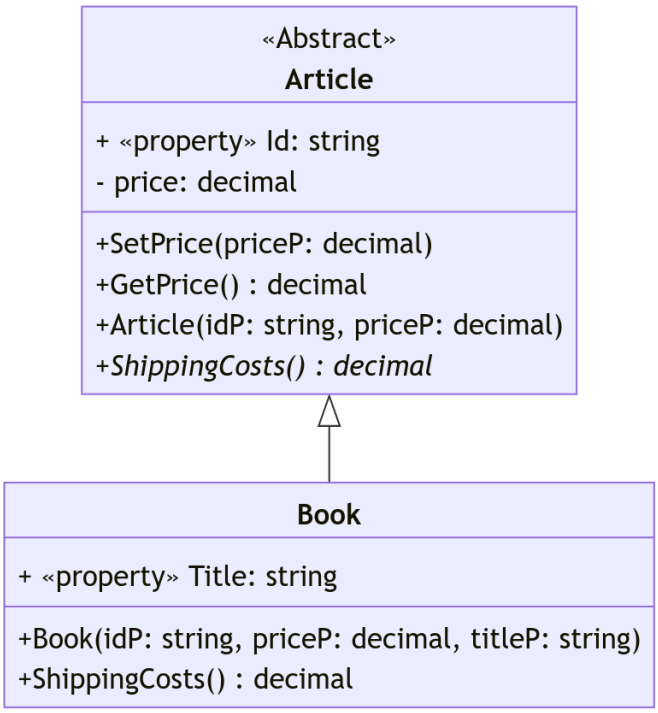
2. Write a program that displays "Your passcode is sorted" if `passcode` is sorted (either in ascending or descending order).

3. Write a program that displays the number of matching characters in the same position in both arrays. For example, if `passcode` contains `b`, `C`, `4`, `a` and `codeentered` contains `B`, `4`, `c`, `a`, then "1" should be displayed, as only one character (`a`) is in both arrays at the same position.

4. Write a program that displays the number of matching characters in both arrays. For example, if `passcode` and `codeentered` are as in the previous question, then "2" should be displayed, as two characters (`4` and `a`) are in both arrays[1].

[1] Assume all the characters in `passcode` are different, and the same for `codeentered`: `a`, `a`, `b`, `b` would not be valid.

____ / 50 pts. **Problem 3** Consider the following diagram:



1. Write a (partial) implementation of the `Article` abstract class:

   (a) Write an implementation for the `price` attribute: you can either use a getter and a setter (as pictured in the UML diagram), or a property. However, in both cases, setting the price to a negative value should result in an `ArgumentOutOfRangeException` (that you can shorten to `AOORE`) exception being thrown.

   (b) Write an *abstract* `ShippingCosts()` method.

2. Now, assume given a complete implementation of the `Article` abstract class. Write a **complete** implementation of the `Book` class (header included), containing:
   (a) An implementation of the `Title` property using auto-properties.
   (b) A `Book` constructor that passes the `idP` and `priceP` arguments to the `Article` constructor. The `titleP` argument should be assigned to the `Title` property.
   (c) A `ShippingCosts()` method that returns either 5.0, or 10% of the Book's price, whichever is smallest.

3. Write statements that, if placed in a `Main` method, would
   (a) Create a `Book` with `Id` "AAA001", `price` $12.5, titled "What it's like to be a bird".
   (b) Display (nicely) its shipping costs.
   (c) Display its `Id` (as retrieved from the object).

____ / 20 pts. **Problem 4** Write the `Swap` and `Init` methods (header included, as if they were part of the `Main` method) so that the snippets on the left would display what is given in the right.

1. `Swap` exchanges the value of its two arguments.

```
int x = 4, y = 3;
Swap(ref x, ref y);
C.WL($"x is {x}, y is {y}.");
```

Output window:

```
x is 3, y is 4.
```

2. `Init` stores inside of its `char` parameter the first letter of its `string` parameter.

```
char fLetter;
Init("Test", out fLetter);
C.WL("Test starts with "+
 ↪  fLetter);
```

Output window:

```
Test starts with T
```

____ / 15 pts. **Problem 5** Suppose given a `string` variable called `test` that was already initialized, write statements that would display it, reversed, if it is not **`null`** nor empty, and would display `llun` otherwise. For example, if `test` contains `Test123`, then `321tseT` would be displayed.

____ / **35 pts.** **Problem 6** This problem is about 2-dimensional arrays.

1. Write a statement declaring an array of `int` called `matrix` containing the following:

```
12 15 6
 1  3 8
```

2. Write a series of statement displaying the `matrix` array as in the previous question.

3. Write a series of statement displaying the total of each row of the `matrix` array. For our example, it would display "33" and "12".

4. Write a series of statement displaying the total of each column of the `matrix` array. For our example, it would display "13", "18" and "14".

5. Write a series of statement that would create a 2-dimensional array containing the "mirror" of the `matrix` array. In our example, the array created would contain

```
8 3 1
6 15 12
```

Instructions: This exam is to be taken in silence, without notes, books, or electronic devices (including "smart" watches or earbuds). The time limit to complete it is 2 hours. Answer the following questions and problems, trying to be as clear and as accurate as possible, **and remembering that a partial answer is better than no answer at all**. Take your time to read the statements carefully before trying to answer them. If you need more space, write on the back of your test page and indicate it clearly. When writing code, make sure your special punctuation characters are legible, and your lowercase and uppercase letters are easy to distinguish. As usual, every statement or series of statement is assumed to be in a valid class and method, and you can use the `C.RL()`, `C.W()` and `C.WL()` abbreviations.

Program Execution Examples: On the problem below, we provide an example program execution in `this font` (typewriter), underlining the user input and representing "return" with ←. You do not need to reproduce it exactly; it is just to illustrate the expected behavior.

```
This is a prompt by the program.
This is what the user enters.←
```

Problem 1 Write (on the first page) a program that

1. Asks the user to enter a path,
2. Checks if there is a file at this path:
   (a) If there is a file, display its content at the screen.
   (b) If there is no file, ask the user to enter text. When the user enters exactly "!DONE!", on a single line, without the quotes, write the text entered before !DONE! into a file located at the given path.

Your program should be able to handle graciously possible issues (such as an invalid path). Below are two examples of execution, taking place one after the other.

Example execution #1

```
Enter a path
/home/user/CSCI_1302/final/test.txt←
Now creating a file at /home/user/CSCI_1302/final/test.txt.
Enter your text, one line at a time. When done, type "!DONE!" (without the
quotes), then enter.
This is a test←
spanning over←
three lines.←
!DONE!←
File correctly written.
```

Example execution #2

This execution takes place *after* the program was executed as in example execution #1 above.

```
Enter a path
/home/user/CSCI_1302/final/test.txt←
Now displaying file at /home/user/CSCI_1302/final/test.txt.
This is a test
spanning over
three lines.
Done displaying file.
```