Module: COMP1551	Resit Coursework
Application Development	Contribution: 100% of the grade
Module leader: Konstantin Kapinchev	Due date:
Approximate time to complete the coursework: 50 hours	
Learning outcomes: 1. Demonstrate detailed understanding of established industrial standards by writing software requirements specifications for object-oriented, data-driven and interactive applications. 2. Design object-oriented, data-driven and interactive applications by using UML. 3. Develop and test object-oriented, data-driven and interactive applications by applying object-oriented principles, such as encapsulation, abstraction, inheritance and polymorphism. 4. Apply advanced object-oriented techniques to develop advanced software solutions, which solve real-world problems.	

Coursework Submission Requirements:

- An electronic copy of the coursework is expected to be submitted on Moodle before the deadline

- Submit Part 1 as a single C# source code file (.cs)

- Submit both Part 2 and Part 3 as a single PDF file

- The last uploaded version will be the one that is marked

- The limit of the file size is 100 MB

- The submitted documents should be virus-free, not protected by a password or corrupted

- The source code needs to be selectable text, otherwise it will not be marked

- The coursework cannot be submitted after the deadline or via email

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See https://www.gre.ac.uk/student-services/regulations-and-policies for details.

Use of AI

In case AI-generated content is included in the coursework, it needs to be enclosed in quotation marks and properly referenced. This applies to both text and source code. Otherwise, if detected, it may be considered plagiarism.

Coursework Specification:

Part 1

Develop a **C# Windows-based console application**, which has the following class (in addition to the class containing the Main method):

class StringProcessing

The class is expected to contain the following methods:

Constructor

This method initialises an input string S of maximum length of 40 characters of capital letters only and an integer value N with values between -25 and 25. Utilise the C# feature Property to implement data validation. Display an error message, if the user provides characters other than capital letters, or if N is initialised with values outside the interval [-25 .. 25]. In such cases, prompt the user to re-enter the input string S, and/or the integer value N. Repeat this process until the correct data is provided.

Method Encode

This method transforms the input string S into an output string by replacing every character from input string S with a new character. The new character is N places away in the ASCII table from the character entered by the user. The direction is determined by the sign of the integer value N. If the end of the alphabet is reached, continue at the beginning if necessary (wrap around the alphabet). For example, if N = 3, input string S = "ABC" will be transformed into output string "DEF" and if N = -3, input string S = "ABC" will be transformed into output string "XYZ".

Method Print

This method returns the output string generated as a result of the method "Encode".

Method InputCode

This method returns an array of integers. The integers are the ASCII codes, or ASCII values, of the characters from the input string.

Method OutputCode

This method returns an array of integers. The integers are the ASCII codes, or ASCII values, of the characters from the output string.

Method Sort

This method sorts the input string S alphabetically into a new string and returns the newly generated string.

General Notes:

- When developing the methods, use encapsulation to ensure consistency and security of the data. Do not use input and output operations, such as ReadLine and WriteLine, in the class methods. Use those operations in the Main method of the program.

- In the Main method, test the class by instantiating it and invoking all of its methods. Provide input string S and integer value N. Record the output. In a separate test, provide the output string as an input string S and change the sign of the N value. The result is expected to be the same as the initially entered string.

- The program is expected to run on any standard C# compiler.

Part 2

Based on the program developed in **Part 1**, write a ***software requirements specification*** of maximum 500 words. The ***software requirements specification*** is expected to have the following content:

1. Introduction
    - Purpose
    - Project scope
2. Overall Description
    - Product
    - Users
    - Operational Environment
3. System Features
    - Description
    - Functional Requirements
4. User Interface Requirements
5. Platform Requirements
6. Quality Attributes
    - Performance
    - Security
    - Safety

For more information about writing software requirements specification, refer to:

Wiegers, Beatty, "Software Requirements", Third Edition, Microsoft:

Chapter 10 Documenting the Requirements, page 181

Chapter 11 Writing Excellent Requirements, page 203

Part 3

Provide the following **UML diagrams**, which represent the design of the program developed in **Part 1**:

- Class Diagram
- Use Case Diagram

Add captions describing the diagrams.

For more information about UML, refer to Patrick Grassle "UML 2.0 in Action".

Grading criteria

70-100% All requirements completed to an excellent standard.

60-69% All requirements completed. However, there are a number of minor deficiencies in significant areas.

50-59% All requirements completed. However, significant improvements could be made in many areas.

40-49% All requirements completed. However, significant improvements could be made in all areas.

30-39% All requirements attempted but the overall level of understanding and performance is poor.

0-29% There are requirements missing or completed to a very inadequate standard, which indicates a very poor or non-existent level of understanding.

Grading Components

Task	**Maximum points**
Part 1	
Constructor correctly implemented	15
Method Encode correctly implemented	35
Method Print correctly implemented	5
Method InputCode correctly implemented	5
Method OutputCode correctly implemented	5
Method Sort correctly implemented	5
Utilise data validation via Property	5
Utilise encapsulation	5
All methods are successfully tested in Main method	10
Part 2	
Software Requirements Specification is provided as described	5
Task 3	
UML Class Diagram and Use Case Diagram are included	5