* **Section I: Introduction**

Implement software solutions to the following tasks using the Python programming language. Follow the Python Code Convention and document the code with relevant comments where necessary. Test your program/code using appropriate test cases.

Each question should be submitted in a separate Python file. i.e. Question1.py, Question2.py, Question3.ipynb and Question4.py.

All files and functions in the file must be given docstrings describing what you have done, and code should be suitably formatted. Failure to comply will result in loss of marks.

Also, you need to submit one word document that include the source codes of all questions with two screenshots of the testing results (test at least two scenarios for each of the tasks.

Upon completion, submit your solutions to NILE using the submission procedure in Section IV below. Please note that access to only teaching materials, syntax references and other resources posted on NILE is permitted/encouraged.

You are not allowed to make use of regular expressions and / or make use of any classes such as Pattern to determine the String expression. Only the imports that we request you to provide are allowed, NO OTHER imports can be included.

**Using an AI-generated model tool to provide answers is considered academic misconduct. We have internal tools to detect if an answer has been generated by an AI tool. The University of Northampton's policy on academic misconduct and plagiarism will be strictly enforced.**

**Question 1: [60 marks]**

Your task is to create a customer application, where a customer can order books. Once an order has been placed, the order needs to be shipped. Therefore, a shipping cost must be determined. Once complete, an invoice for the order, including the shipping cost, needs to be generated. For example, a customer can buy a book; and the book shop will then place the order. The shipping for the order must be calculated and a shipping order generated. Finally, an invoice must be produced and all invoices added to a repository. The repository can be used to search for an invoice. Further detail regarding each class is now discussed.

**Customer class: [5 marks] the Customer class consists of customer objects**

* + The attributes are as follows: name, phone and eMailAddress (all String);
  + The init() constructor: pass in 3 variables (name, phone, eMailAddress) as parameters to the constructor;
  + The getter() methods: each attribute must be provided with a getter(); - You can use property() function or @property decorator if you like.

As an example, the init () constructor can be like:

def \_\_init\_\_(self, name: str, phone: str, email: str):

  self.name = name

  self.phone = phone

  self.email = email

**Stock class: [5 marks] - the Stock class consists of book objects**

* + The attributes are as follows: book\_name and author (both String), and a price (double);
  + The init() constructor: pass in 3 variables (book\_name, author, price) as parameters to the constructor;
  + The getter() methods: each attribute must be provided with a getter(); - You can use property() function or @property decorator if you like.

**Order class: [5 marks] - the Order class consists of order objects**

* + The attributes are as follows: customer (object of Customer class) and a book i.e. stock (object of Stock class);
  + The init() constructor: pass a customer object and a stock object to the constructor;
  + The getter() methods: each attribute must be provided with a getter(); - You can use property() function or @property decorator if you like.

As an example, the init () constructor can be like:

def \_\_init\_\_(self, customer: Customer, stock: Stock):

  self.customer = customer

  self.stock = stock

**Shipping class: [15 marks] the Shipping class sets the shipping cost and creates a shipping order**

* + import: datetime package;
  + The attributes are as follows: an order object (Order) , a ship\_date (LocalDate), a ship\_cost (double) and count\_urgent (int);
  + The init() constructor: pass an order object and a ship\_date to the constructor;
  + The getter() methods: the attributes ship\_date and ship\_cost require a getter();- You can use property() function or @property decorator if you like.
  + The set\_ship\_cost() method: pass the ship\_cost in as a parameter, set the parameter to the attribute ship\_cost;
  + The calc\_ship\_cost () method: pass a boolean parameter is\_urgent to the method, to determine the ship\_cost make use of the is\_urgent value; depending on whether the shipping is urgent or not the shipping cost will differ, urgent shipping means that the ship\_cost is set to 5.45, otherwise if it is not urgent set the ship\_cost to 3.95, if the shipping is urgent increment the count\_urgent, return the ship\_cost;

**Invoice class: [5 marks] - the Invoice class generates an invoice for the order placed**

* + The attributes are as follows: invoice\_nbr (String), the book order, i.e. stock (Stock) object, ship\_order (Shipping) object and the total\_cost (double);
  + The init() constructor: pass an invoice\_nbr, stock object, and the ship\_order object to the constructor;
  + The getter () method: the attribute invoice\_nbr requires a getter(); - You can use property() function or @property decorator if you like.
  + The invoice() method: calculate the total\_cost for shipping this order which would be the price of the book (stock) and the shipping cost, return the total\_cost;

**BookStore class: [15 marks] create a list (repository) to keep all invoices as well as searching for an invoice**

* + The attributes are as follows: declare the list invoices, final string message is provided to you;
  + The init() constructor: complete the declaration of the list invoices (instantiate);
  + The getter () method: get\_invoices(); - You can use property() function or @property decorator if you like.
  + The search\_invoice() method: pass a string parameter nbr to the method, retrieve an invoice from the repository where the invoice number equals the nbr parameter, print out the message "Invoice not found" if the invoice does not exist;

*[*OPTIONAL*] You can also use a Python library called collections to use specialized data structures like deque or Counter, which can be useful in certain scenarios.*

**Test class: [10 marks] - the Test class tests the classes**

The main method must do the following:

* + create 3 objects of the Customer class
  + create 3 objects of the Stock class
  + create 3 objects of the Order class
  + create 3 objects of the Shipping class
  + create 3 objects of the Invoice class
  + create 1 object of the BookStore class
  + add the invoices to the repository
  + search for an invoice

You can use the following class Test as a template for your testing:

class Test:

    def main():

        customer1 = #continue your code here

        customer2 = #continue your code here

        customer3 = #continue your code here

        stock1 = #continue your code here

        stock2 = #continue your code here

        stock3 = #continue your code here

        order1 = Order(customer1, stock1)

        order2 = #continue your code here

        order3 = #continue your code here

        shipping1 = Shipping(order1, datetime.date.today())

        shipping2 = #continue your code here

        shipping3 = #continue your code here

        shipping1.set\_ship\_cost(shipping1.calc\_ship\_cost(True))

        shipping2.set\_ship\_cost(shipping2.calc\_ship\_cost(False))

        shipping3.set\_ship\_cost(shipping3.calc\_ship\_cost(True))

        invoice1 = Invoice("INV0001", stock1, shipping1)

        invoice2 = #continue your code here

        invoice3 = #continue your code here

        bookstore = BookStore()

        bookstore.invoices.append(invoice1)

        bookstore.invoices.append(invoice2)

        bookstore.invoices.append(invoice3)

        print(f"Number of urgent shipments: {shipping1.count\_urgent}")

        print(f"Invoice 1 total cost: {invoice1.invoice():.2f}")

        print(f"Invoice 2 total cost: {invoice2.invoice():.2f}")

        print(f"Invoice 3 total cost: {invoice3.invoice():.2f}")

        bookstore.search\_invoice("INV0004")

if \_\_name\_\_ == "\_\_main\_\_":

    Test.main()

*Your code needs to be well commented and documented. You must submit all classes in one file called****Question1.py****. Also, in the word document submission, you must list your code and take screenshots of all test cases.*

* **Question 2 [20 Marks]**

Write a program to analyse the sentiment of a piece of text (whether a text conveys a positive or negative opinion). The program should consist of two classes: SentimentLexicon and Classifier.

The SentimentLexicon class should have a dictionary instance variable that is initialized in the constructor. It should also have a method that takes in two parameters: paths to following positive and negative word list files.

**positive-words.txt**

**negative-words.txt**

Each line of each file should be read and stored in the dictionary instance variable as follows: if the word is from the positive word list file, it should have a value of 1 in the dictionary; if the word is from the negative word list file, it should have a value of -1.

The Classifier class should have an instance variable that is an instantiated SentimentLexicon object. It should also have a method called classify that takes in a single parameter: a text string. For each word in the string, this method should look up the word in the sentiment lexicon and calculate a score by summing the values for each word. If the overall score is positive, the classify method should return a value of 1. If the overall score is negative, it should return a value of -1. If the score is 0, it should return 0 (indicating no sentiment).

To test the program, create a main method that does the following:

* + Creates a Classifier object
  + Iteratively classifies each of the following sentences, leaving the punctuation in:
  + "I love Python."
  + "Python is the language I love!"
  + "The iPhone is clearly not the most terrible and worst phone ever. It is the best."

For each sentence that is classified, store the results in a dictionary as follows and print them out:

* + {'text': 'I love Python', 'sentiment': 1}
  + {'text': 'Python is the language I love!', 'sentiment': 1}
  + {'text': 'The iPhone is clearly not the most terrible and worst phone ever. It is the best.', 'sentiment': 1}

*Your code needs to be well commented and documented. You must submit a file called****Question2.py****. Also, in the word document submission, you must list your code and take screenshots of all test cases.*

* **Question 3 [8 marks]**

This Question will test your knowledge of NumPy or Pandas and how to program using this package and as such you should use the appropriate functionality provided by only NumPy or Pandas and no other external libraries. You should write your answer in a python notebook i.e, dot ipynb file. You feel free to use any kind of jupyter notebook version (anaconda or google colab). Note: it's recommended to use google colab in case you have any installation issues with anaconda.

The following information is an estimate of the population distribution across the United Kingdom along with the area of the country:

* + - England has a population of 55,550,138 and an area of 130,309 square kilometres.
  + - Scotland has a population of 5,466,000 and an area of 77,911 square kilometres.
  + - Wales has a population of 3,169,586 and an area of 20,736 square kilometres.
  + - Northern Ireland has a population of 1,895,510 and an area of 13,793 square kilometres.

Create two dictionaries, for population and area. Use these dictionaries to create a `DataFrame` object and print the resulting `DataFrame`, with a column named 'population' as the first column and a column named 'area' as the second column. Print the resulting `DataFrame`.

*Your code needs to be well commented and documented. You must submit a file called****Question3.ipynb****. Also, in the word document submission, you must list your code and take screenshots of all test cases.*

* **Question 4 [12 marks]**

This Question will test your knowledge of NumPy or Pandas and how to program using this package and as such you should use the appropriate functionality provided by only NumPy or Pandas and no other external libraries.

* + This is the creative part of the assignment and is worth 12 marks. In the following file `food-data.xlsx` containing information on typical food a student may eat. Using pandas to read in and work with the data, present 6 interesting statisics or claims that you can find by using pandas functionality. Each finding you present should be different in nature if possible. Explain, with comments, how you have reached each of the findings and why it is interesting. N.B - do not manipulate the source file (`food-data.xlsx`) itself.
  + In addition to the above task, you are also required to create a GUI (graphical user interface) using Tkinter to present your findings in a visual way. This can be in the form of graphs, charts, or other visualizations. Make sure to include explanations of your visualizations and how they relate to your findings. You can add various widgets to it to create the desired layout for your GUI. For example, you might add a label widget to display a title for your visualisation, or a button widget to allow the user to interact with your GUI. To create the visualisation themselves, you can use a module such as matplotlib to generate the plots and charts, and then use a widget such as a Canvas widget in Tkinter to display the visualisation in your GUI. Finally, you will need to include explanations of your visualisation and how they relate to your findings. This can be done using label widgets or other types of text widgets to display the explanations.

**food-data.xlsx**

*Your code needs to be well commented and documented. You must submit a file called****Question4.py****. Also, in the word document submission, you must list your code and take screenshots of all test cases.*

* **Section IV: Submission Procedure**

1. Copy and paste all your answers and code into a WORD document. Then take a screenshot of the console output after running at least two test cases. Submit this to the word document dropping point.

2. The folder containing all your Python programs must be zipped and uploaded to NILE, e.g. *johnSmith.zip*where John Smith is your student name. Submit this to the zip file dropping point.

Please keep in mind that it is your obligation to submit the correct files to the appropriate submission links.

* + Please submit the word document to the word document submission link (NOT to the zip file link).
  + Please submit the zip file to the zip file submission link (NOT to the word document link).
  + Otherwise, your submissions will be marked as ZERO