* **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. Task1.py, Task2.py, Task3.py and Task4.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.**

* **Section II: Extensions for Welfare Reasons**

Only the Mitigating Circumstances team are allowed to grant extensions, so please do not ask any of your lecturers or Teaching Assistants (for any module) for an extension. You can talk to Mitigating Circumstances Team in confidence about any problems and they do not tell us the reason for granting you an extension.

* **Section III: Tasks**

**Task 1: [10 marks]**

A checksum is a single number that can act as a kind of “digital signature” of a long string. Just like how two people will have different handwritten signatures, two strings can have two different digital signatures. There are many ways to calculate the checksum of any arbitrary string; the more complex the calculation, the less likely it is for two strings to have the same checksum value.

Following the function design recipe, complete the function that takes a string of characters as input and returns the checksum. The checksum for this program should simply be the sum of the ord() values of each character, modulo 10. For example, the checksum for the word “cat” would be (99 + 97 + 116) % 10 which is 2.

*To submit this, you must submit a file called task1.py. Also, in the word document submission, you must list your code and take screenshots of two test cases.*

* **Task 2: [20 marks]**

You’ve been hired to help write the software for an automatic change dispenser, the kind you see on a self-checkout machine at a supermarket, that automatically dispenses the right coins depending on the amount of change that’s owed to the customer. Following the function design recipe, complete the function that takes in a number from 0 to 99 (representing the amount of change that’s due) as a parameter and then returns, in the following order as individual variables, the number of one penny, two pence, 5 pence, 10 pence, 20 pence, and 50 pence pieces that should be dispensed. Your function should be as efficient as possible and dispense the fewest coins possible.

*To submit this, you must submit a file called task2.py. Also, in the word document submission, you must list your code and take screenshots of two test cases.*

* **Task 3: [40 marks]**

Using only the the attached text file (unmodified) for development and testing, write a program to make sense of London Marathon data to meet the following specifications (tip: carefully look through the file to ensure you understand how it is structured before developing your code):

**marathon\_London.txt**

* + Read in the file. The name of the file should be taken as input from the user.
  + Write a function that returns the number of lines in the file, excluding the header. This function should be named lineCount.
  + Write a function that given a name as a parameter, will return the country that the runner is representing. This function should be named runnerCountry.
  + Write a function that given a country name as a parameter, will return a list containing all rows containing information about runners from that country. This function should be named countryInformation.
  + Write a function that calculates and returns as a string the average time that the runners from the input file took to complete the London Marathon. This function should be named averageTime.

*To submit this, you must submit a file called task3.py. Also, in the word document submission, you must list your code and take screenshots of two test cases.*

* **Task 4: [30 marks]**

A farmer is trying to determine the percentage of his crops that will be affected by pests. In order to do this, he takes a sample of 1000 randomly selected plants, and counts the number of those plants that have been affected by pests. For example, let's say that out of the 1000 plants that were sampled, 220 of them have been affected by pests. Therefore, the sample proportion of crops affected by pests is 220/1000 = 0.22, or 22%.

Using a Monte Carlo simulation, the goal is to use this sample proportion as a starting point, and then use random sampling to estimate the probability that more than 20% of the farmer's entire crop will be affected by pests. In other words, the simulation aims to estimate the probability that the proportion of affected plants in the entire crop is greater than 0.2 (20%).

The program should use a random number generator to simulate the sampling process multiple times (a large number, for example 10,000) and for each simulation, it should randomly select a sample of 1000 plants from the entire crop, and calculate the proportion of affected plants in that sample. Then, it should record whether the proportion of affected plants in that sample is greater than 20%. After all the simulations are completed, it should calculate the percentage of times that the proportion of affected plants was greater than 20% , and compare it to the actual proportion derived from the sample data. It would be expected that the result will be close to 22% (sample proportion). Use the top-down design strategy to develop this. Also, briefly describe your logic design at the top of your python file as a multiline comment on the top of your python file.

*To submit this, you must submit a file called task4.py. Also, in the word document submission, you must list your code and take screenshots of two 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
* **Note:**

1. The specification given for each task (e.g. program name, method name, etc.) and all the submission guidelines must be strictly followed. If not, penalties will be imposed.

2. The University of Northampton Policy on Academic Misconduct and Plagiarism will be strictly implemented. You may be asked for an online viva-voce should there be any doubts in the originality of the submitted work.