CS2201: Introduction to Computation
End Semester : Monday Group (21/04/2025 10:15AM)

Duration: 2 hours 30 mins **Total marks: 40**

Instructions

- All programs should be compatible with Python 3.
- There will be a plagiarism check. The suspected copies will attract a deduction of marks for the problem irrespective of who the original author of the code is. Copying from the internet will also be considered as plagiarism. However, you may reuse the codes shared by the instructor.
- Late submissions are not allowed.
- You are advised to compile the code before submitting it to WeLearn. If a code does not compile at our end, marks will be deducted for the problem.
- Each program should follow a strict naming convention: **QNo.py** (e.g. Q1.py, Q2a.py etc.). Programs not adhering to the convention will not be considered for evaluation.
- All codes (.py files) should be submitted in a single zipped folder named **YourWeLearnID.zip** (e.g. 23MS123.zip) to WeLearn.
- You should put appropriate comments in the code without which marks will be deducted.

1. **(Marks: 10)** Consider the following table (Table 1) showing the amount (in million kgs) of Darjeeling tea produced in West Bengal in the last few years.

Year	2016	2018	2020	2022	2024
Production	6.5	6.9	6.7	6.6	6.5

Table 1: Darjeeling Tea Production: Q1

Implement an appropriate Interpolation method in Python to estimate the Darjeeling tea production in the year 2017.

2. **(Marks: 10)** A private bank **BankOnUs** is looking to identify *Churn* tendency (inclination of customers to switch to a competitor bank) by segregating such customers from the rest based on two features: number of transactions and balance. Note that, the customers looking to switch to another bank are expected to perform a low number of transactions and maintain a low account balance at the current bank. Consider a training dataset (which you can use) in Table 2 shows some finance expert identified churn tendencies in some customers of the same bank.

   Take the details (the number of transactions and balance) of a new customer as input from the user. Using the kNN classification algorithm (choose k appropriately) and the training data in Table 2, determine if the new customer is likely to opt for another bank. If the customer is classified as *churn*, print a message: *We have an exciting offer for you! Would you like to know more?*; else print a message: *Thank you for banking with us! Please let us know how we can serve you better, on the bank website.*

3. **(Marks: 10)** In a secret chemical lab **ChemLab**, scientists must authenticate themselves each time they enter the lab. An automatic authenticator bot asks for a *code* (unique identifier of a scientist) and after the scientist provides the same (e.g. *code* is *A071*), the authenticator bot looks up a table (created by you) that stores *code*: *original passcode* pairs. If the code matches one of the codes in the database

Customer ID	No. of Transactions	Balance (INR)	Churn
1	44	870	1
2	40	600	1
3	41	790	1
4	50	900	1
5	1000	100000	0
6	1500	350000	0
7	2000	250000	0
8	1000	150000	0
9	25	1001	1
10	50	500	1

Table 2: Training data: Q2. The value 1 in "Churn" indicates the tendency for that customer to switch to a competitor bank.

(as a second level of security check), the ciphered passcode pc_c is asked. For example, if for the *code* as *A071*, the original passcode pc_o is *FOX*, the correct ciphered passcode pc_c is *CLU*. This ciphering is done according to the scheme in Table 3. The ciphered version of the original letter set is obtained by `rotating` the former *left* by k (length of the passcode). That is, the **Ori** $\rightarrow$ **Cph** mapping is *not* static as it changes if the length of the passcode is different. E.g. if the original passcode is *PAULING*, the value of k will be 7, and so the left rotation will be by 7 places.

If the scientist provides the correct ciphered passcode, he/she gets the entry. Implement this scheme in Python by first taking the *code* as input. If it is correct, go to the next level (otherwise, deny access). At the next level, ask for the ciphered version of the *passcode*, allowing entry in case of correct match, else deny entry. A welcome message should be on the screen, followed by the questions. Each allow or denial should be indicated by printing an appropriate message on the screen. An example sequence of successful entry is as below:

Welcome to ChemLab!!
Please Input your CODE:
Input: A071
Please enter your ciphered PASSCODE:
Input: CLU
Welcome, Dr. FOX!!

Ori	**A**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cph	X	Y	Z	**A**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Table 3: Cipher: Q3. Ori: Original, Cph: Ciphered for the Original text *FOX* (length = 3)

4. **(Marks: 10)** The Large Language Models (LLMs) are known to apply next token prediction schemes based on pre-training on sequences of text. Consider the training text (after preprocessing; delimited by the full-stops): *harry love to eat . he is a good boy . he read a lot . but he is a food lover . he eat chocolate . he eat candy . he eat pizza . he eat chocolate again . he eat chocolate every day . he is never tired of eat .*

   The probability p_{i+1} of the next token t_{i+1} is dependent on the current token (t_i) and the previous tokens; can be expressed as $P(T = t_{i+1}|T = t_iT = t_{i-1}...)$. If we consider a bigram probability, we consider only the current token, so that $p_{i+1} = P(T = t_{i+1}|T = t_i)$. From the training dataset, given the token "he" (t_i), the next token can be estimated from all the words that have appeared *after* "he" from multiple sentences (separated by the full-stops) – "is", "read", and "eat". However, "eat" will be the most probable choice because it appears 5 out of 9 times (estimated probability of $5/9 \approx 0.56$) after "he". *Note that the estimation can be done within a sentence and not across sentences.* The sentences are separated by the full-stops.

Using the above scheme (for bigram), implement an algorithm in Python that asks the user for token τ after which the next token is to be predicted. If the input token τ is present in the training data, print the most probable next token; else, print a message saying that the prediction cannot be done. The training data can be pre-stored by you based on which you may perform the estimations. Note that next token prediction is also not possible for the tokens that *do not* have a following word (within the sentence) in the training text.