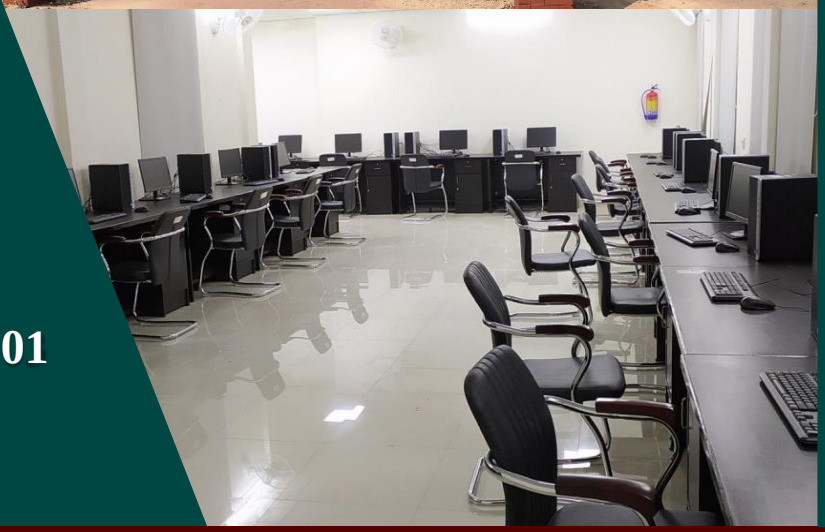
2024-25

جب کوئی قوم فن اور علم سے عاری ہو جاتی ہے تو وہ غربت کو دعوت دیتی ہے اور جب غربت آتی ہے تو وہ ہزاروں جرائم کو جنم دیتی ہے۔

"When a nation becomes devoid of art and learning, it invites poverty and when poverty comes it brings in its wake thousands of crimes."
-Sir Syed Ahmad Khan



MCA III Semester Lab Manual

Lab-III

Course Code- CAMS3P01

DEPARTMENT OF COMPUTER SCIENCE

ALIGARH MUSLIM UNIVERSITY, ALIGARH

2024-2025

Credits

The following lab manual up-gradation committee updated the lab manual:

- ***Prof. Arman Rasool Faridi (Chairperson)***
- ***Prof. Mohammad Ubaidullah Bokhari***
- ***Prof. Aasim Zafar***
- ***Prof. Suhel Mustajab***
- ***Dr. Faisal Anwar***
- ***Dr. Mohammad Sajid***
- ***Dr. Mohammad Nadeem***
- ***Dr. Faraz Masood***

The following committee member originally design the lab manual:

- ***Prof. Mohammad Ubaidullah Bokhari***
- ***Dr. Arman Rasool Faridi***
- ***Dr. Faisal Anwer***
- ***Prof. Aasim Zafar (Convener)***

Design & Compilation:

- ***Dr. Faraz Masood***

Revised Edition: July, 2024

Department of Computer Science, A.M.U.,

Aligarh (U.P.) India

Lab Manual: Lab – I (CAMS-3P01)

COURSE TITLE: Lab-I **COURSE CODE:** CAMS-3P01

CREDIT: 2 **PERIODS PER WEEK:** 6

CONTINUOUS ASSESSMENT: 60 **EXAMS:** 40

COURSE DESCRIPTION

Application of classroom knowledge and skills in computer science to solve real-world problems and to develop research and software development skills.

CONTENT

This course consists of the development of a realistic application, representative of a typical real-life software system, under semi-professional working conditions. The students are expected to propose, analyses, design, develop, test and implement a software system. The student will deliver oral presentations, progress reports, anda final report.

Depending on the topic of the project and the chosen software development methodology, which may vary from one year to another, the following themes may be addressed to some extent:

- Software development methodologies, static (products) and dynamic aspects (processes);
- Requirement analysis (goals, use cases), software architectures, architectural styles and patterns, model-driven engineering (MDE);
- Programming techniques, software development environments, refactoring;
- Software validation through unit tests, integration tests, functional and structural tests, and code reviews.
- Project management, planning, resource estimation, reporting.
- Version management by using a version management tool.
- Examples of kinds of systems to be developed are distributed systems, client/server systems, web-based systems, secure systems, mobile systems,

adaptable systems, optimizations of existing systems or data-intensive systems, etc.

Besides completing a mini project, the students are required to complete subject related Lab Assignments given by respective course teachers. The individual teachers who are teaching the courses with lab component will be responsible for giving assignments, monitoring and evaluating their respective assignments.

OBJECTIVES

- To help students develop openness to new ideas in computer science, develop the ability to draw reasonable inferences from observations and learn to formulate and solve new computer science problems using analytical and problem-solving skills;
- To help students develop the ability to synthesize and integrate information and ideas, develop the ability to think creatively, develop the ability to think holistically and develop the ability to distinguish between facts and opinion;
- To help students acquire the necessary competences to build a real-life software system by completing different software life cycle phases (like, specification, architecture, design, implementation, validation, documentation, etc);
- To help students develop the ability to work individually and as part of a team, develop a commitment to accurate work, develop management skills, improve speaking and writing skills, improve the ability to follow directions, instructions and plans, and improve the ability to organize and use time effectively.
- To help students develop a commitment to personal achievement, the ability to work skilfully, informed understanding of the role of science and technology, a lifelong love of learning, and cultivates a sense of

responsibility for one's own behaviour and improves self-esteem/self-confidence.

OUTCOMES

Upon successful completion of this course students will be able to:

- ❒ Identify project/research problems; understand information and grasp meaning; translate knowledge into new context; use information, methods, concepts, and theories of fundamental topics in computer science in new situations (*Knowledge*, *Comprehension*);
- ❒ Apply computer science principles and practices to a real-world problem; demonstrate in-depth knowledge in the area of the project they have undertaken; solve problems using required knowledge and skills; implement and test solutions/algorithms (*Application and Evaluation*);
- ❒ Identify potential solutions/algorithms for the project problem; see patterns and modularize the problem, recognize hidden meanings and identify components, show proficiency in software engineering principles (*Analysis*);
- ❒ Apply a software development methodology currently practiced in industry to produce software system in a rigorous and systematic way using different software life cycle phases (specification, architecture, design, implementation, validation, documentation) (*Synthesis*);
- ❒ Show evidence (group collaboration, regular meetings, email communications, significant knowledge and skills contributions, etc.) of working productively as an individual and in a team on a project that produces a significant software product (*Team Work*);
- ❒ Show evidence of competency in oral and written communications skills through oral presentations (project presentation, department seminar or

conferences, client interactions), technical reports and/or published research papers in conferences and/or journals (*Communications*);

- ❒ Use modern techniques, skills and tools necessary for computer science practices relevant to the project they undertake; use techniques in recent research papers to solve problems (*Lifelong Learning*).

HOW TO DO WELL IN THIS COURSE

- ❒ The students are advised to attend all their theory classes and respective labs regularly as both are integrated to each other. If any student will miss the theory lecture, he/she may not able to do well in lab related to that topic.
- ❒ The students are advised to submit the assignments given in theory and lab classes timely to their respective Teachers/Instructors.
- ❒ The students should demonstrate disciplined and well behaved demeanor in the Department.
- ❒ Each student shall be assigned a system in their introductory lab. They are advised to do their work on that system only for the whole semester. Students should store all their lab activities regularly.
- ❒ All students are advised to understand course objectives and outcomes and achieve both during their lab work.
- ❒ The students are advised to follow books/eBooks/online tutorial/other online study material links given in lecture/lab manual/ syllabus references. These study materials are very helpful in terms of skills, knowledge and placement.
- ❒ This lab course is very important in terms of placement. Therefore, students are advised to implement all the problems by her/him given in the individual week.

- All students are advised to solve old placement papers for campus selection. Following links may be useful for the preparation of your campus placements.
  - https://www.indiabix.com/placement-papers/companies/
  - https://www.offcampusjobs4u.com/download-tcs-placement-test-question-papers-with-solutions/
  - https://www.indiabix.com/placement-papers/tcs/
  - https://www.firstnaukri.com/career-guidance/infosys-placement-papers-with-solutions-2019-firstnaukri-prep
  - https://prepinsta.com/ibm/
  - https://www.faceprep.in/infosys/infosys-aptitude-questions/
  - https://alpingi.com/infosys-placement-papers-solution-pdf-download/
  - http://placement.freshersworld.com/
- The Students are advised to follow below tutorials' links:
  - https://www.w3schools.com/php/php_intro.asp
  - https://www.w3schools.com/js/default.asp
  - https://www.w3schools.com/sql/default.asp
  - https://www.tutorialspoint.com/php/index.htm
  - https://www.geeksforgeeks.org/php/
  - https://www.geeksforgeeks.org/sql-tutorial/
  - https://www.geeksforgeeks.org/javascript-tutorial/
- The Students are advised to follow below Links for installing application software:
  - https://www.eclipse.org/pdt/
  - http://php.net/manual/en/install.php
  - https://docs.microsoft.com/en-us/dotnet/framework/install/guide-for-developers
  - https://httpd.apache.org/download.cgi

- https://docs.microsoft.com/en-us/sql/database-engine/install-windows/install-sql-server?view=sql-server-ver15
- https://www.sqlservertutorial.net/install-sql-server/

❒ The Students are advised to follow below Links for online compilers :

- https://www.onlinegdb.com/online_php_interpreter
- http://phpfiddle.org/
- https://www.tutorialspoint.com/execute_sql_online.php
- https://www.onlinegdb.com/online_sqlite_editor
- https://www.onlinegdb.com/online_csharp_compiler

❒ Skill Set that are required to develop by the students of MCA:

- Good communication and behavioral skills
- A positive attitude
- Confidence
- Strong technical skills
- Good command over programming languages like C, C++, Java, .Net, etc.
- Good programming skills and hands on experience
- Knowledge of data structure and database
- Awareness of latest technology trends.

TEACHING METHODS ANDASSESMENTS FOR ACHIEVING LEARNING OUTCOMES

This lab class will meet thrice per week for 100 minutes each meeting -- some lab class time may be traditional lectures, reviewing concepts and tools that are useful for the mini project, but most lab class time will be used for guided discussion and development, student presentations, and some team meetings. However, some lab classes may be used to discuss and solve subject related lab assignments given by the respective teachers of courses.

Generally, students will be taking up mini projects individually. However, in situations when they are working in teams, the individual responsibilities should be planned and documented throughout the phases of the project.

Students are expected to choose an appropriate project topic in consultation with the teacher, and do a short presentation that "pitches" the idea to the teacher and the class. While there is some flexibility in project selection, students should keep in mind the "capstone" nature of this class. Students must develop projects that demonstrate that they have a working knowledge of basic and advanced concepts in computer science and also demonstrate a reasonable knowledge of recent developments in computer science. Each project should include non-trivial software development that has been approved by the teacher/instructor.

With an approved project, students will proceed through a standard sequence of software development stages, beginning with a requirements analysis and specification, and concluding with a final evaluation. A complete detail of the all project stages is given in the milestones section as well as summarized in "TOPICAL CALENDAR" section. At the end of each stage, each individual/team must produce a written report giving stage-specific documentation and describing the work performed, problems encountered, and decisions made. For team projects, the report must include a meeting log and breakdown of tasks by team member. One week before the completion of each stage, there will be a presentation from each project that previews the progress and results in that phase, for in-class discussion and suggestions for refinement in the following week. For these intermediate stage presentations, team members will rotate through as "presenter" for the team, and each student must make at least two intermediate-stage presentations (for a 3-person team this means that there will have to be multiple presentations on the same stage).

In the case of a group project, each member of the group must present the entire project, highlighting their individual contributions toward the project's success, and

a short summary of each individual's contributions should be included in the final report as well.

SUBMISSION OF DELIVERABLES

Final project report, including all the deliverables, is required to be submitted strictly as per notified schedule.

EVALUATION AND GRADING

Students work on a single project throughout the duration of this course, and their course grade is calculated based on the grades for individual aspects and milestones. The project will be graded for completeness, content, correctness, quality of presentation (oral and written reports), team work (in case of group project), and the demonstration of the student's knowledge in the computer science field.

As per the University norms Mini Project Report shall be finally evaluated by the external examiner at the end of the semester. However, there will be continuous monitoring of the progress and evaluation of the Mini Project during the semester and the distribution of marks shall be as follows:

Proposal	**5 %**
Presentations 1 & 2 **Presentations-1: Proposal/Synopsis** **Presentations-II: SRS & Design**	**15 %**
Progress Report 1 **Requirements/Specification and Planning and Analysis**	**20 %**
Progress Report 2 **System/Research Design**	**10 %**
Progress Report 3	**10 %**

Implementation and Testing

Final Deliverables:

Final Presentation	**10 %**
Technical Report (including final source code)	**30 %**

REQUIRED TEXTS/READING/REFERENCES

Readings and references are project-specific, and will be determined by students/project groups, with approval of the teacher. All the resources used should be properly referenced.

Students will be making extensive use of external references for their project, and should be vigilant in maintaining high standards with regard to attribution and avoidance of plagiarism. If there are questions about how to deal with any such matters, the student should discuss the matter with the teacher concerned to make sure there are no misunderstandings.

ATTENDANCE POLICY

Attendance is vital for this class, since discussions, regular oral presentations and progress reports will have a strong impact on the ability to complete the project. You may be dropped from the course for missing more than two consecutive scheduled meetings/presentations.

LATE POLICY

Late work will not be accepted. In case of any unavoidable situations, make requests with the teacher/instructor to reschedule the assigned work/task on case to case basis, if possible.

MILESTONES FOR MINI-PROJECT

1. ***Deciding and Registering the topic/title of the mini-project***

   All the students are required to decide the topic/title for ***real life software project***, which they want to design, develop and implement. In finalizing the proposed work topic, they may take help from concerned teachers/mentor in the lab. Decided topic/title ***needs to be approved*** by the concerned teacher/mentor in the lab.

   Parallel Activity: Keep preparing the brief summary (synopsis/proposal) of the proposed work as per the given format.

2. ***Submission of brief summary (synopsis/proposal) of the proposed work (As Per the Prescribed Format)***

   Once the project topic/title is decided and approved, all students are required to **submit** and **present** a brief summary of the proposed work (synopsis/proposal), clearly specifying the client's requirements for which the application software is being developed and the main features of the proposed software. After incorporating the suggestions of the teachers, if any, the final version of the summary of the proposed work (synopsis/proposal) should be submitted to concerned teacher in the lab.

3. ***Submission Requirement specifications, planning and analysis***

   After the submission and presentation of summary of the proposed work (synopsis/proposal), the students need to submit progress report-1, which includes: analysis modeling and related diagrams (please refer to the Topical calendar for more insights).

4. ***System Analysis and Design Phase (Submission of SRS document)***

   All students are required to study and analyze the present system (or existing manual system or proposed system), and all the findings should be **submitted** and **presented** in the form of ***SRS document*** along with ***gantt chart***

(using appropriate gantt tool, like GanttProject). While doing so, they may actively be involved with client, and/or teachers/Mentors for discussion. Some of the templates/formats of the typical ***SRS document*** are being attached for your reference.

Refer for SRS template:

- http://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
- www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc

Discuss the ***SRS document*** with the concerned teacher/Mentor in the lab, incorporate suggestions (if any), and maintain the different versions of SRS document. Students are required to present and submit *the final signed version of* ***SRS document*** to the concerned teacher/Mentor in the lab.

SRS document should contain the ER diagram, Data Flow Diagram and Data Dictionary. You should also prepare important UML diagrams like Use Case diagrams, Activity diagrams, class diagrams, behaviour model and/or state transition diagram etc.

Students are advised to use standard tools for drawing UML diagrams, DFD, ER Diagrams, etc. Examples of some typical tools are listed below:

- UML diagrams using automated tool such as StarUML, BOUML etc.
- Data Flow Diagram (DFD) with different levels using tools such as Lucidchart, Visual Paradigm, etc.
- E-R Diagram with the help of automated tools such as ERDPlus, Smartdraw, etc.

Parallel Activity: In the mean time, you may learn and practice the tools necessary to develop the proposed software, and finalize the detailed database design, including populated tables. Also students should start the coding in parallel with their presentation of ***SRS document***.

5. System Development (Coding/Testing)

Start the development of the system as per the design specifications discussed in *SRS document*. Coding should be well documented. ***Technical Report*** specifying the brief technical specifications and documenting the working of each major modules/methods should be submitted. Students are required to properly maintain the following during the system development:

- A clear design of working database of the system using a popular DBMS such as ORACLE, SQL Server, MYSQL, etc.

Deployment/Implementation

Deploy/Implement the developed system on the client site (actual user site) and ***obtain*** user acceptance letter, specifying that the developed system is working satisfactorily and is as per the specified requirements. Better you prepare an installation copy for your software along with installation manual.

Parallel Activity: Keep writing and preparing the ***Final project Report*** as per the standard format. (***Refer:*** Format for Project Report.pdf)

6. Final Evaluation

Demonstrate the working of system to the audience (teacher, Mentor students, clients), specifying the design and main features of the developed system. The **final project report** and an **oral presentation** should be submitted as per standard project report format/template. The user manual must be a part of the final project report and should be written in explanatory manner so that anyone can operate the system using this manual. Hardcopy as well as softcopy of the all the reports (like SRS Document, Technical report, Final Project report) should be submitted to the concerned teacher/Mentor in the lab. Softcopy of complete project (code), database and necessary files (preferably, installation software, along with installation manual) should also be submitted (***Refer:*** Format for Project Report.pdf).

TOPICAL CALENDAR				
S. No.	**Project Stage/Activities**	**Deliverables**	**Duration**	**Deadline**
01	**Deciding and Registering the Topics/Titles of the Mini project**	**Registration of the Project topic/title Parallel Activity:** Synopsis preparation	1-2 Week	17th Aug 2024
02	**Brief Summary (Synopsis/Proposal) of the proposed work**	**Submission of Preliminary** Proposal/Synopsis	2-3 Week	31st Aug 2024
		Presentation 1 (Proposal/Synopsis) **Parallel Activity:** Requirement Specifications and Planning and Analysis	3-4 Week	
03	**Requirement Specification, Planning and Analysis**	**Progress Report 1 Submission** • Approach and System Profile • Uses Cases • Feasibility and Draft Model • System and algorithm analysis • Preliminary object/process model • Tool selection, etc.	5-7 Week	21st Sept 2024
04	**System/Research Design**	**Progress Report 1 Submission(SRS & Design Document)**	7-8 Week	5th Oct 2024
		Presentation 2 (SRS & Design) **Parallel Activity:** Implementation, Deployment and Testing	8-9 Week	
05	**Implementation, Deployment and Testing**	**Progress Report 2 Submission (Technical Report**) • Includes brief code walkthrough • Source code • Test results and discussion	9-12 Week	5th Nov 2024
06	**Evaluation and Refinement**	**Presentation 3 (Final presentation)Parallel Activity:** Evaluation and Refinement (Final Report)	13-14 Week	
		Final Report submission	14 Week	16th Nov 2024

Note*: *The last date of submission of various activities can be changed and notified separately from time to time.

LAB ASSIGNMENTS

APPENDIX–I

Template for the Index of Lab File

WEEK NO.	PROBLEMS WITH DESCRIPTION		PAGE NO.	SIGNATURE OF THE TEACHER WITH DATE
1	1#			
	2#			
	3#			
2	1#			
	2#			
	3#			
3	1#			
	2#			
	3#			

Note: The students should use Header and Footer mentioning their roll no. & name in footer and page no in header.

WEEK #1

OBJECTIVES

- ❒ To practice basic Python programming concepts.
- ❒ To solve simple mathematical and logical problems using Python.

Outcomes

After completing this week, the students would be able to:

- ❒ Write Python programs for basic arithmetic operations.
- ❒ Implement basic algorithms for checking conditions and iterating through lists.

Problems

1. Write a program to find the product of two user-supplied integers and if the product is equal to or lower than 5000, then return the sum of the two numbers.
2. Write a program to print the sum of the first 10 numbers.
3. Write a program to iterate through a supplied list of 20 numbers and print only those numbers which are divisible by 5.
4. Write a program to check if the given number is a palindrome.
5. Write a program to calculate the cube of all numbers from 1 to a given number.

WEEK #2

Objectives

- ❒ To understand the use of loops in Python.
- ❒ To perform digit manipulation and basic mathematical operations using loops.

Outcomes

After completing this week, the students would be able to:

- ❒ Extract digits from numbers and perform calculations.
- ❒ Use loops to iterate through numbers and lists.

Problems

1. Write a program to extract each digit from an integer in reverse order.
2. Write a program to count the total number of digits in a number using a while loop.
3. Write a program to display all prime numbers within a range.
4. Write a program to use a loop to find the factorial of a given number.
5. Write a program to find the sum of the digits of a supplied integer.

WEEK #3

Objectives

- To learn pattern printing using loops in Python.
- To perform basic string manipulations and indexing.

Outcomes

After completing this week, the students would be able to:

- Print various patterns using loops.
- Manipulate strings and perform operations based on string indexes.

Problems

1. Write a program to print the following pattern using the for loop:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

2. Write a program to print the following star pattern using the for loop:

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

3. Write a program to print characters from a string which are present at even index numbers.
4. Write a program to accept a string from the user and display characters that are present at even index numbers.
5. Write a program to remove characters from a string starting from the nth position to the last and return a new string. Example: remove_chars("aligarh", 3) should output ali.

WEEK #4

Objectives

- ❒ To understand the creation and use of functions in Python.
- ❒ To manipulate lists and perform operations on list elements.

Outcomes

After completing this week, the students would be able to:

- ❒ Create and use functions to perform specific tasks.
- ❒ Manipulate list elements and perform operations like iteration and transformation.

Problems

1. Write a program to create a function cal_sum_sub() that accepts two variables and calculates addition and subtraction. Also, it must return both addition and subtraction in a single return call.
2. Write a function to return True if the first and last number of a given list are the same. If the numbers are different, return False.
3. Given a list of numbers, write a program to turn every item of the list into its square.
4. Given two Python lists, write a program to iterate both lists simultaneously and display items from list 1 in original order and items from list 2 in reverse order.
5. Write a program to count the number of occurrences of item 50 in the tuple tp1 = (50, 10, 60, 70, 50).

WEEK #5

Objectives

- To generate random data and perform operations on it.
- To manipulate numpy arrays and perform sorting operations.

Outcomes

After completing this week, the students would be able to:

- Generate random data like OTPs and passwords.
- Perform operations on numpy arrays and sort them.

Problems

1. Write a program to generate a 6-digit random secure OTP.
2. Write a program to pick a random character from a user-supplied string.
3. Write a program to generate a random password that meets the following conditions:
    a. Password length must be 10 characters long.
    b. It must contain at least 2 uppercase letters, 1 digit, and 1 special symbol.
4. Given two lists of numbers, write a program to create a new list containing odd numbers from the first list and even numbers from the second list.
5. Write a program to create a numpy array and return an array of odd rows and even columns from the numpy array.
6. Write a program to create a numpy array and sort it as per the following cases:
    a. Case 1: Sort the array by the second row.
    b. Case 2: Sort the array by the second column.

WEEK #6

Objectives

- To perform operations on tuples and dictionaries in Python.
- To manipulate data stored in tuples and dictionaries.

Outcomes

After completing this week, the students would be able to:

- Combine and manipulate tuples.
- Create and manipulate dictionaries.

Problems

1. Write a Python program that inputs two tuples and creates a third that contains all elements of the first followed by all elements of the second.
2. Write a Python program to create a dictionary with names and phone numbers. Then ask the user for a name and print the corresponding phone number.
3. Write a Python program to calculate the sum of squares of the first two digits and the last two digits of a 4-digit number, e.g., for 1233, calculate 12^2 + 33^2.
4. Write a program that inputs a main string and creates an encrypted string by embedding a short symbol-based string after each character. The program should also decrypt the string.
5. Write a program to get roll numbers, names, and marks of students and store these details in a file called "Marks. data".
6. Write a program to accept a string and display:
    a. Number of uppercase characters
    b. Number of lowercase characters
    c. Total number of alphabets
    d. Number of digits

WEEK #7

OBJECTIVES

- ❒ To learn about different Linux shells and their types.
- ❒ To understand file handling utilities in Linux.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Differentiate between various Linux shells (sh, csh, ksh, bash).
- ❒ Use file handling utilities to manage files in Linux.
- ❒ Understand Data Science

PROBLEMS

1. Write a report on different types of Linux shells (sh, csh, ksh, bash).
2. Copy, move, and remove files using cp, mv, and rm commands.
3. Create and delete directories using mkdir and rmdir.
4. Change the current working directory using cd and display the present working directory using pwd.
5. Consider two files that contain information about Employees and Departments in the following parameters: Employee (Name, EId, Salary, DID), Department (DID, DName, DLocation). Write a Python program to find the average salary of each department.
6. Consider two files having some lines of statements. Write a Python program to swap content present at middle line of first file with the content of last line of the second file. (Note: First file contains odd numbers of lines of statement)

WEEK #8

OBJECTIVES

- ❒ To understand basic file attributes and permissions in Linux.
- ❒ To learn how to manage file and directory permissions.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Display and interpret file attributes using ls options.
- ❒ Change file and directory ownership and permissions.
- ❒ Understand Data Science

PROBLEMS

1. Display detailed information about files using ls -l and other ls options.
2. Change file ownership using chown and group ownership using chgrp.
3. Modify file and directory permissions using chmod.
4. Demonstrate how to set and remove directory permissions.
5. Write a python program that reads a string which contains English alphabets and numbers. The program should create two lists L1 and L2, where L1 includes all the numbers present in the string while L2 includes all the alphabets of the string.
6. Write a program in python to find vowels having maximum number of instances in a given file.

   (Note: File contains variety of data such as English alphabets, numbers etc.).
7. Write a Python program to create a list of user's supplied distinct integers having even number of elements. The program further creates two lists of equal lengths based on the original list, where first list is having all elements less than elements of second list. Display both the lists.

WEEK #9

OBJECTIVES

- ❒ To understand the Linux file system and the concept of inodes.
- ❒ To learn how to manage file systems and inodes.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Describe the structure of the Linux file system.
- ❒ Explain the role of inodes in the Linux file system.
- ❒ Understand Data Science.

PROBLEMS

1. Write a report on the structure of the Linux file system.
2. Display inode information using ls -i and interpret the results.
3. Create and delete files and directories, and observe changes in inode numbers.
4. Explain the significance of inodes in file management and demonstrate with examples.
5. Write a program in python to find alphabet/s having maximum number of instances in a given file.
6. A file contains information about programs and courses in the following format: Program,course. Write a Python program to find the number of courses against each program.

Eg:
Program,Course
MCA,Database
MCA,Java
M.Sc,Data Structure
B.Sc, Python

The output should be: MCA-2, M.Sc-01, B.Sc-01

7. A file contains information about employees with the following parameters: Name, Id, Salary, Dname. Write a Python program to write one more column HRA (House rent allowances) to this file, where HRA= 18%of salary

   Eg: Suppose the existing file is as follows, where you need to add HRA column:

Name,id,salary, Dname
Amar,101,20000,Sales
Ammar,102,22000,Marketing
Rahil,103,18000,Sales

WEEK #10

OBJECTIVES

- ❒ To learn the basics of shell programming using Bash.
- ❒ To write simple shell scripts to automate tasks.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Write and execute basic shell scripts.
- ❒ Use variables, loops, and conditionals in shell scripts.
- ❒ Understand Data Science

PROBLEMS

1. Write a simple shell script to display "Hello, World!" on the terminal.
2. Write a shell script to accept user input and display it.
3. Write a shell script to demonstrate the use of variables.
4. Write a shell script to perform basic arithmetic operations.
5. Write a program in python to find word/s having maximum number of instances in a given file and replace all its occurrences with “Aligarh”.
6. Consider two files that contain information about Employees and Departments in the following parameters: Employee (Name, EId, Salary, DID), Department (DID, DName, DLocation). Write a Python program to merge the content of both the file in following format.: Emp_Dep(Ename, Eid, Esalary, EDID, DName,Dlocation) (Note: Merging should follow the condition-DID of Employee file should be equal to Department ID of department file)

WEEK #11

OBJECTIVES

- ❒ To learn advanced shell scripting techniques.
- ❒ To write more complex shell scripts using loops, conditionals, and functions.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Implement loops and conditionals in shell scripts.
- ❒ Create and use functions in shell scripts.
- ❒ Understand Data Science

PROBLEMS

1. Write a shell script to check if a file exists and display an appropriate message.
2. Write a shell script to find the factorial of a number using loops.
3. Write a shell script to demonstrate the use of conditionals (if-else statements).
4. Write a shell script to create and use a simple function.
5. You have been given a dataset demo.csv having independent features as x1, x2, x3, x4, x5, x6, x7 and dependent feature as y with value either 0 or 1. All independent features are continuous data except x1 and x2, which are having nominal data. Now write python program for the following:
    a. Clean independent features
    b. Add one more feature x7 having values between 0 and 1.
    c. Perform scaling
    d. Train this dataset using Logistic regression, Decision Tree and Random Forest. Compare the performance of all the models based on accuracy and F1 score.
    e. Draw confusion matrix of each model

WEEK #12

OBJECTIVES

- ❒ To apply shell scripting knowledge to solve practical problems.
- ❒ To write shell scripts for common system administration tasks.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Write shell scripts to automate common tasks.
- ❒ Apply shell scripting to manage files, directories, and processes.
- ❒ Understand Data Science

PROBLEMS

1. Write a shell script to back up a directory to a specified location.
2. Write a shell script to monitor disk usage and send an alert if usage exceeds a threshold.
3. Write a shell script to automate the creation of user accounts.
4. Write a shell script to search for a specific pattern in a file and display the results.
5. Consider two features x and y based on the following function:

   $y = x1^2 + 3x2 + c$, where c can be prepared based on 1000 random values between 0 and 1

   Now generate 1000 random values between 0 and 1 for x1 and x2. Calculate y based on above function. Now train Polynomial Regression model and check the score for the same.

WEEK #13

OBJECTIVES

- ❒ To understand the architecture and features of Linux.
- ❒ To learn the basics of the vi editor and common Linux commands.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Describe the architecture and salient features of Linux.
- ❒ Use the vi editor to create and edit files.
- ❒ Execute basic Linux commands.
- ❒ Understand Data Science

PROBLEMS

1. Write a report on the architecture and features of Linux.
2. Create and edit a text file using the vi editor.
3. Execute basic Linux commands like pwd, cd, ls, mkdir, and rmdir.
4. Create a text file in the vi editor, make changes, and save it.
5. Consider MNIST dataset that contains 70,000 small square 28×28-pixel grayscale images of handwritten single digits between 0 and 9. Train MNIST dataset using any model that you studied to classify a given image into digit 8 (Binary classification problem). Discuss its performance also.

WEEK #14

OBJECTIVES

- ❒ To develop comprehensive shell scripting projects.
- ❒ To integrate various shell scripting techniques and commands into larger scripts.
- ❒ To learn data science through practice problems.

OUTCOMES

After completing this, the students would be able to:

- ❒ Develop complex shell scripts to solve real-world problems.
- ❒ Integrate multiple shell commands and scripting techniques into cohesive projects.
- ❒ Understand Data Science

PROBLEMS

1. Develop a shell script to automate the setup of a web server, including installing necessary packages and configuring the server.
2. Write a shell script to perform a system health check, including checking CPU usage, memory usage, and disk space, and generate a report.
3. Create a shell script to manage user permissions and automate the backup of critical files.
4. Write a comprehensive shell script to manage and rotate system logs, ensuring old logs are archived and deleted after a certain period.
5. You have been given a dataset temp.csv having independent features as x1,x2,x3,x4,x5 and dependent feature as y with value either 0 or 1. All independent features are continuous data except x5, which is having nominal data. Now write python program for the following:
    a. Clean independent features (if any)
    b. Draw heatmap to show correlations among independent features.
    c. Train this dataset using Logistic regression, Decision Tree and Random Forest. Compare the performance of all the models based on accuracy and F1 score.

d. Draw confusion matrix of each model
e. Check whether scaling improves the performance or not.

6. Consider two features x and y based on the following function:

   y = 3x + k, where k can be prepared based on 1000 random values

   Now generate 1000 random values between 0 and 1 for x. Calculate y based on above function for these 1000 values of x. Now train Linear Regression model and check the score for the same.