Department of Computer Science

Lab Manual

Go Lang

MCA372D

Master of Computer Applications

2024-25

Prepared by:
Dr.Binayak Dutta , Dr Deepa V Jose

Verified by: HoD

Department Overview

Department of Computer Science of CHRIST (Deemed to be University) strives to shape outstanding computer professionals with ethical and human values to reshape nation's destiny. The training imparted aims to prepare young minds for the challenging opportunities in the IT industry with a global awareness rooted in the Indian soil, nourished and supported by experts in the field.

Vision

The Department of Computer Science endeavours to imbibe the vision of the University "**Excellence and Service"**. The department is committed to this philosophy which pervades every aspect and functioning of the department.

Mission

"To develop IT professionals with ethical and human values". To accomplish our mission, the department encourages students to apply their acquired knowledge and skills towards professional achievements in their career. The department also moulds the students to be socially responsible and ethically sound.

Introduction to the Programme

Master of Computer Applications (MCA) is a post-graduate programme of CHRIST (Deemed to be University) and approved by the All India Council of Technical Education (AICTE). It is a two-year programme spread over six trimesters to bridge the chasm between computer studies and applications bringing within reach of enthusiastic youngsters an excellent group of experienced and dedicated staff and experts, and an enviable infrastructure.

MCA at CHRIST (Deemed to be University) strives to shape outstanding computer professionals with ethical and human values to reshape nation's destiny. The training programme aims to prepare young minds for the challenging opportunities in the IT industry with a global awareness rooted in the Indian soil, nourished and supported by experts in the field. The Department is committed to the motto "Excellence and Service" and this philosophy pervades every aspect of its functioning.

Programme Objective

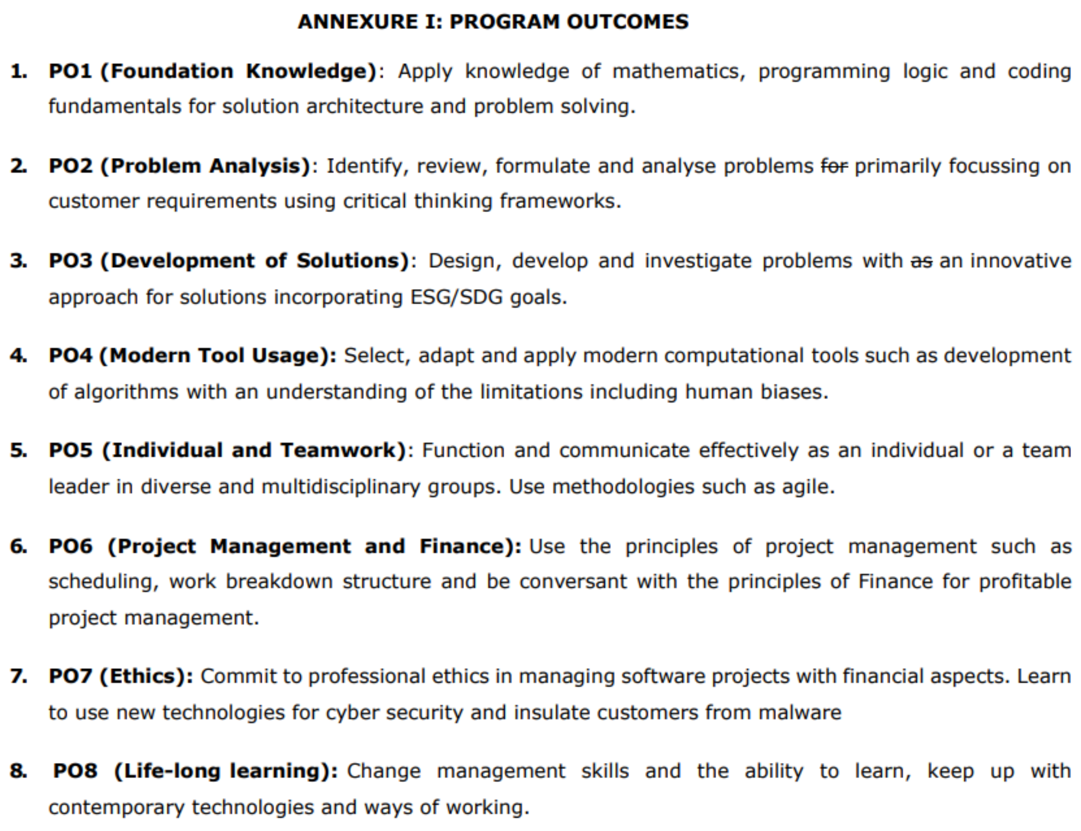
This programme is conceptualised and designed based on the strong commitment of the department to provide better quality education to the students. The principal objectives of this course are to:

- Heighten technological awareness
- Train future industry specialists
- Encourage effective software development
- Provide research training
- Involve students in innovative research
- Produce graduates with a two-year professional education in Computer Science with technical, professional, and communications skills.

Ethics and Human Values

1. Only proprietary or open-source software would be used for academic teaching and learning purposes.
2. Copying of programs from internet, friends or from other sources is strictly discouraged since it impairs development of programming skills.
3. Unique Practical (Domain based) exercises ensures that the students don't involve in code plagiarism.
4. Projects undertaken by students during the course are done in teams to improve collaborative work and synergy between team members.
5. Projects involve modularization which initiates students to take individual responsibility for common goals.
6. Passion for excellence is promoted among the students, be it in software development or project documentation.
7. Giving due credit to sources during the seminar and research assignment is promoted among the students
8. The course and its design enforce the practice of good referencing technique to improve the sense of integrity.
9. Courses involving group discussions and debates on ethical practices and human values are designed to sensitize the students in dealing with customers and members within the organization.

Programme Outcomes:



ANNEXURE I: PROGRAM OUTCOMES

1. **PO1 (Foundation Knowledge)**: Apply knowledge of mathematics, programming logic and coding fundamentals for solution architecture and problem solving.
2. **PO2 (Problem Analysis)**: Identify, review, formulate and analyse problems ~~for~~ primarily focussing on customer requirements using critical thinking frameworks.
3. **PO3 (Development of Solutions)**: Design, develop and investigate problems with ~~as~~ an innovative approach for solutions incorporating ESG/SDG goals.
4. **PO4 (Modern Tool Usage):** Select, adapt and apply modern computational tools such as development of algorithms with an understanding of the limitations including human biases.
5. **PO5 (Individual and Teamwork)**: Function and communicate effectively as an individual or a team leader in diverse and multidisciplinary groups. Use methodologies such as agile.
6. **PO6 (Project Management and Finance):** Use the principles of project management such as scheduling, work breakdown structure and be conversant with the principles of Finance for profitable project management.
7. **PO7 (Ethics):** Commit to professional ethics in managing software projects with financial aspects. Learn to use new technologies for cyber security and insulate customers from malware
8. **PO8 (Life-long learning):** Change management skills and the ability to learn, keep up with contemporary technologies and ways of working.

LIST OF PROGRAMS

1. Implement the concept of Variables, values and type.

2.Implement the concept of control flow.

3. Implement the concept of Array and Slice.

4. Implement the concept of Map and Structs.

5. Implement the concept of functions and error handling

6. Implement the concept of interface

7. Implement the concept of Pointers, call by value and call by function.

8. Implement the concept of JSON marshal and unmarshal. Write its unit test case.

9. Implement the concept of Concurrency.

10. Implement the concept Goroutines and Channels

Program 1: Variables, Values, and Types

Details of the Lab Exercise Create a Go program that demonstrates the declaration and initialization of variables of various types including integers, floats, strings, and booleans.

Scenario/Case Study Develop a program to store details of a product: product name (string), price (float), stock quantity (integer), and availability (boolean).

Pseudo Code

1. Declare variables for product details.
2. Initialize them with appropriate values.
3. Print the values and their types using `fmt.Printf`.

Test Cases/Expected Output

- Input: Hardcoded values
- Output: Properly formatted product details with types.

Evaluation Rubrics

- Correctness of variable declaration (3 points)
- Appropriate use of types (2 points)
- Clarity of output (1 point)

Mapping of RBT

- Apply (Level 3)

Mapping of CO

- CO1, CO3

References

https://go.dev/ref/spec#Variables

Program 2: Control Flow

Details of the Lab Exercise Write a Go program to demonstrate `if`, `for`, and `switch` statements by calculating the factorial of a given number.

Scenario/Case Study Create a program to check if a number is prime, calculate its factorial, and categorize it as even or odd.

Pseudo Code

1. Input a number.
2. Check if it's prime using a loop.
3. Calculate factorial using a `for` loop.
4. Use a `switch` statement to categorize the number.

Test Cases/Expected Output

- Input: 5
- Output: Prime, factorial 120, odd

Evaluation Rubrics

- Use of control flow constructs (5 points)
- Correctness of calculations (3 points)
- Clarity of output (2 points)

Mapping of RBT

- Apply (Level 3)

Mapping of CO

- CO1, CO2

References

- https://go.dev/ref/spec#If_statements

Program 3: Arrays and Slices

Details of the Lab Exercise Demonstrate the use of arrays and slices by finding the maximum and minimum values from a list of integers.

Scenario/Case Study Create a program to analyze daily temperatures recorded over a week.

Pseudo Code

```
1. Declare an array for 7 temperatures.
2. Convert it to a slice.
3. Iterate through the slice to find max and min.
```

Test Cases/Expected Output

- Input: [30, 25, 28, 35, 20, 31, 29]
- Output: Max: 35, Min: 20

Evaluation Rubrics

- Proper use of arrays and slices (5 points)
- Correctness of calculations (3 points)
- Readability (2 points)

Mapping of RBT

- Analyze (Level 4)

Mapping of CO

- CO1, CO2

References

- https://go.dev/ref/spec#Array_types
- https://go.dev/ref/spec#Making_slices_maps_and_channels

Program 4: Maps and Structs

Details of the Lab Exercise Create a program to store and manipulate data using maps and structs.

Scenario/Case Study Develop an address book application to store names and contact numbers.

Pseudo Code

1. Define a `struct` for contact details.
2. Use a map to store name-contact pairs.
3. Implement CRUD operations.

Test Cases/Expected Output

- Input: Add, view, delete contact
- Output: Correct operations

Evaluation Rubrics

- Struct and map usage (5 points)
- Functionality (3 points)
- Code readability (2 points)

Mapping of RBT

- Create (Level 6)

Mapping of CO

- CO2, CO3

References

- https://go.dev/ref/spec#Types

Program 5: Functions and Error Handling

Details of the Lab Exercise Implement a program to divide two numbers, with proper error handling for division by zero.

Scenario/Case Study Create a calculator program with basic operations and error handling.

Pseudo Code

1. Define functions for addition, subtraction, multiplication, and division.
2. Use `error` to handle invalid cases.

Test Cases/Expected Output

- Input: Division by zero
- Output: "Error: Division by zero"

Evaluation Rubrics

- Function definition (4 points)
- Error handling (4 points)
- User experience (2 points)

Mapping of RBT

- Apply (Level 3)

Mapping of CO

- CO1, CO2

References

- https://go.dev/ref/spec#Built-in_functions
- https://go.dev/ref/spec#Errors

Program 6: Interfaces

Details of the Lab Exercise Create a Go program to demonstrate the use of interfaces by implementing geometric shapes.

Scenario/Case Study Calculate the area and perimeter of circles and rectangles using interfaces.

Pseudo Code

1. Define an interface for `Shape`.
2. Implement methods for `Circle` and `Rectangle`.
3. Use polymorphism to calculate properties.

Test Cases/Expected Output

- Input: Circle radius 3, Rectangle 4x5
- Output: Circle area 28.27, Rectangle area 20

Evaluation Rubrics

- Interface usage (5 points)
- Code reusability (3 points)
- Correctness (2 points)

Mapping of RBT

- Create (Level 6)

Mapping of CO

- CO1, CO2

References

- https://go.dev/ref/spec#Interface_types

Program 7: Pointers

Details of the Lab Exercise Demonstrate pointers by swapping two variables.

Scenario/Case Study Create a function to swap values using pointers.

Pseudo Code

```
1. Define variables.
2. Pass pointers to a function.
3. Swap values in the function.
```

Test Cases/Expected Output

- Input: x=5, y=10
- Output: x=10, y=5

Evaluation Rubrics

- Correct pointer usage (5 points)
- Functionality (3 points)
- Clarity (2 points)

Mapping of RBT

- Apply (Level 3)

Mapping of CO

- CO1, CO2

References

- https://go.dev/ref/spec#Pointer_types

Program 8: JSON Marshalling/Unmarshalling

Details of the Lab Exercise Create a program to serialize and deserialize user data using JSON.

Scenario/Case Study Develop a program to save and load user profiles.

Pseudo Code

1. Define a `struct` for user profiles.
2. Marshal data to JSON.
3. Unmarshal JSON back to Go.

Test Cases/Expected Output

- Input: Hardcoded user data
- Output: JSON and Go struct

Evaluation Rubrics

- JSON operations (5 points)
- Unit testing (3 points)
- Clarity (2 points)

Mapping of RBT

- Analyze (Level 4)

Mapping of CO

- CO3

References

- Golang JSON Docs, https://medium.com/@briankworld/working-with-json-data-in-go-a-guide-to-marshalling-and-unmarshalling-78eccb51b115

Program 9: Concurrency

Details of the Lab Exercise Demonstrate concurrency by summing an array using multiple goroutines.

Scenario/Case Study Split an array into parts and calculate the sum concurrently.

Pseudo Code

```
1. Define an array.
2. Use goroutines to sum parts.
3. Combine results.
```

Test Cases/Expected Output

- Input: [1,2,3,4]
- Output: Sum: 10

Evaluation Rubrics

- Concurrency (5 points)
- Performance (3 points)
- Accuracy (2 points)

Mapping of RBT

- Evaluate (Level 5)

Mapping of CO

- CO1, CO3

References

- https://go.dev/blog/pipelines

Program 10: Goroutines and Channels

Details of the Lab Exercise Create a producer-consumer problem using goroutines and channels.

Scenario/Case Study Simulate a task queue where producers add tasks and consumers process them.

Pseudo Code

```
1. Define a channel.
2. Start producer and consumer goroutines.
3. Synchronize using the channel.
```

Test Cases/Expected Output

- Input: Tasks ["A", "B", "C"]
- Output: Processed tasks

Evaluation Rubrics

- Correct goroutine usage (5 points)
- Synchronization (3 points)
- Functionality (2 points)

Mapping of RBT

- Create (Level 6)

Mapping of CO

- CO1, CO3

References

- https://go.dev/tour/concurrency/2