Name of Course/ Module : PROGRAMMING FUNDAMENTALS

Course Code : DFC2073

Name (s) of academic staff :

Rationale for the inclusion of the course/ module in the programme :
This course is continuity to foundational knowledge in problem solving which is a part of the requirement in body of knowledge in Information Technology field. It's a very basic computer programming skills needed to enhance to next level.

Semester and Year offered: Semester 1 / Year 1

Student Learning Time (SLT) | Dependent Learning (DL) | Independent Learning (IDL) | Total
--- | --- | --- | ---
L = Lecture | P = Practical | T = Tutorial | O = Others | 50 | 120
12 | 48 | 0 | 10

Credit value : 3

Prerequisites (if any) : DFC1042 Problem Solving And Program Design

Learning Outcomes :
Upon completion students should be able to:
CLO 1 : explain the fundamental programming constructs element (control structures, arrays, structures, functions and pointers) and articulate how they are used to develop a program. (C2, PLO1)
CLO 2 : apply programming constructs to realise a computer program with debugging techniques to achieve a working program. (C3, P3, PLO1, PLO2)
CLO 3 : solve computing problems using suitable algorithmic solutions and code these algorithmic solutions in a computer programming language. (C4, P3, A2, PLO1, PLO2, PLO4)

Transferable Skills:
Skills and how they are developed and assessed, project and practical experience and Internship
a. Knowledge
b. Practical Skills
c. Critical Thinking and Problem Solving Skills
Skills are assessed through :
Project and Practical Task for Generic Student Attribute (GSA). Knowledge are assessed through theoretical methods (Quiz & Test)

Teaching-Learning and assessment strategy
a. Teaching-Learning Strategy
   Implemented in Problem Based Learning (PBL), guided by lecturers through Face-to-Face and Blended Learning approach.

b. Assessment Strategy
   The course assessment is carried out through Coursework Assessment (CA) and Final Examination (FE).
12 Synopsis

PROGRAMMING FUNDAMENTALS course introduces the fundamental concepts of structured programming, and provides a comprehensive introduction to programming for Information Technology majors. Topics include data types, control structures, functions, arrays, and the mechanics of running, testing, and debugging. Practical lab sessions and problem based questions will help to develop the skills required to identify the best data and program constructs to solve well-defined problems. The course also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers.

13 Mode of Delivery

Interactive Lecture, Discussion, Laboratory Activity, Case Study and Presentation.

14 Assessment Methods and Types

The course assessment is carried out in two sections:

a. Coursework (CA) - 50%

Coursework is continuous assessment that measures knowledge, technical skills and soft skills.

i. Quiz (3) - 10%

ii. Test (1) - 10%

iii. Laboratory Task (4) - 15%

iv. Project (1) - 15%

b. Final Examination (FE) – 50%

15 Mapping of the course/module to the Programme Aims

<table>
<thead>
<tr>
<th>Course Learning Outcome/ Programme Educational Objectives (PEO)</th>
<th>PEO1</th>
<th>PEO2</th>
<th>PEO3</th>
<th>PEO4</th>
<th>PEO5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. Explain the fundamental programming constructs element (control structures, arrays, structures, functions and pointers) and articulate how they are used to develop a program. (C2, PLO1)</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ii. Apply programming constructs to realise a computer program with debugging techniques to achieve a working program. (P3, C3, PLO2, PLO1)</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iii. Solve computing problems using suitable algorithmic solutions and code these algorithmic solutions in a computer programming language. (C4, P3, A2, PLO1, PLO2, PLO4)</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Programme Educational Objectives (PEO)

PEO 1 : Possess relevant knowledge, skills and aptitude to meet job specifications, organisational and system needs;

PEO 2 : Can utilise current computing tools and techniques by applying knowledge and interpreting information to solve problems, can execute and be responsible for routine tasks;

PEO 3 : Have effective communication skills to convey information, problems and solutions;

PEO 4 : Have teamwork and interpersonal skills, entrepreneurial awareness and are aware of their social and ethical responsibilities; and

PEO 5 : Possess skills for lifelong learning and career development.
Mapping of the course/module to the Programme Learning Outcomes

<table>
<thead>
<tr>
<th>Course Learning Outcome (CLO)/Programme Learning Outcomes (PLO)</th>
<th>PLO1</th>
<th>PLO2</th>
<th>PLO3</th>
<th>PLO4</th>
<th>PLO5</th>
<th>PLO6</th>
<th>PLO7</th>
<th>PLO8</th>
<th>PLO9</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. Explain the fundamental programming constructs element (control structures, arrays, structures, functions and pointers) and articulate how they are used to develop a program. (C2, PLO1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ii. Apply programming constructs to realise a computer program with debugging techniques to achieve a working program. (P3, C3, PLO2, PLO1)</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iii. Solve computing problems using suitable algorithmic solutions and code these algorithmic solutions in a computer programming language. (C4, P3, A2, PLO1, PLO2, PLO4)</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Programme Learning Outcomes (PLO)

PLO 1: Apply the foundation of computing, mathematics and soft skills to be competent and possess strong understanding in related Information Technology (IT) fields;

PLO 2: Practice technical skills by applying appropriate methodologies, models and techniques in IT fields;

PLO 3: Communicate effectively with IT Professionals, other professionals and community;

PLO 4: Demonstrate strong analytical and critical thinking skills to troubleshoot and solve problems within realistic constraints by applying knowledge, principles and skills in IT;

PLO 5: Demonstrate an awareness of and consideration for society, health, safety, legal and cultural issues and their consequent responsibilities;

PLO 6: Acquire life-long learning and professional development to enrich knowledge and competencies;

PLO 7: Inculcate entrepreneurial skills in the related discipline that contributes towards national growth and be competitive in IT industries;

PLO 8: Adhere to professional codes of ethics and enhance humanistic values to adapt to the real challenges in working environment; and

PLO 9: Demonstrate effective leadership and teamwork skills.

Content outline of the course/module and the SLT per topic

<table>
<thead>
<tr>
<th>Course Outline (Suggested Sequence of Topics)</th>
<th>Recommended Time Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L</td>
</tr>
<tr>
<td>1.0 INTRODUCTION TO FUNDAMENTALS OF PROGRAMMING</td>
<td>1</td>
</tr>
<tr>
<td>a. C++ program basic structure.</td>
<td></td>
</tr>
<tr>
<td>b. Coding standard guidelines.</td>
<td></td>
</tr>
<tr>
<td>c. Classify identifier and data types.</td>
<td></td>
</tr>
<tr>
<td>d. Rules for framing for an identifier.</td>
<td></td>
</tr>
<tr>
<td>e. Basic of computer program.</td>
<td></td>
</tr>
<tr>
<td>f. Compiling and debugging process, and errors in programming.</td>
<td></td>
</tr>
<tr>
<td>g. Errors in programming</td>
<td></td>
</tr>
<tr>
<td>h. Effective debugging process.</td>
<td></td>
</tr>
</tbody>
</table>
### 2.0 BASIC PROGRAM ELEMENTS
- a. Variables and constants
- b. Data types.
- c. Identifier scope: local, global.
- d. Keywords.
- e. Input output statements.
- f. Operators and expression.
- g. Expression using operators.
- h. Typecasting.
- i. Operators’ precedence.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>0</td>
<td>2.25</td>
<td>8.00</td>
<td>20.25</td>
</tr>
</tbody>
</table>

### 3.0 PROGRAM CONTROL STRUCTURES
- a. Program control structures.
- b. selection control structures.
- c. switch case statement.
- d. break statement.
- e. Loops control structures.
- f. Control transfer statements.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>0</td>
<td>2.5</td>
<td>13.00</td>
<td>30.50</td>
</tr>
</tbody>
</table>

### 4.0 ARRAY, STRUCTURES AND POINTER
- a. The use of arrays.
- b. Two dimensional.
- c. Program using the concept of pointer.
- d. Relationship between pointer and array.
- e. Use of structures

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>0</td>
<td>2.5</td>
<td>13.00</td>
<td>30.50</td>
</tr>
</tbody>
</table>

### 5.0 FUNCTIONS
- a. Functions in programming.
- b. Function prototypes
- c. Call function.
- d. Parameters and arguments.
- e. Scope of variables
- f. Parameters passing techniques.
- g. ‘Pass by value’ and ‘Pass by Reference’.
- h. Passing arrays as arguments to functions.
- i. Call a function to pass an array as argument.
- j. Structures to functions.
- k. Recursive functions.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>0</td>
<td>2.5</td>
<td>11.75</td>
<td>29.25</td>
</tr>
</tbody>
</table>

**TOTAL**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>48</td>
<td>0</td>
<td>10</td>
<td>50.00</td>
<td>120</td>
</tr>
</tbody>
</table>

### 18 Main references supporting the course

### Additional references supporting the course

### Other additional information:
#### Practical session activity
Students perform hands-on activities using C++ Programming software such as Microsoft Visual C++, Borland C++, Turbo C++, or etc. All practical activities **MUST** related to practical activities in Problem Solving and Program Design course.
# Course Syllabus:

## 1.0 INTRODUCTION TO FUNDAMENTALS OF PROGRAMMING

### 1.1 Understand the C++ program basic structure

1.1.1 Describe the item in C++ program structure:
- a. comments
- b. preprocessor directives.
- c. header files.
- d. main() function.
- e. return statements.

1.1.2 Describe two types of comments that supported by C++ program.

1.1.3 Identify the importance of following coding standards.

1.1.4 Apply the coding standard guidelines.

### 1.2 Classify identifier and data types

1.2.1 Explain identifier, variable and constant.

1.2.2 Explain why we need variables.

1.2.3 State the rules for framing for an identifier.

1.2.4 Explain the data types.

1.2.5 List the data types and their respective range with examples:
   - a. Integer
   - b. Floating point
   - c. Character
   - d. Boolean
   - e. String

1.2.6 Name the variables according the standards.

### 1.3 Apply the basic of computer program

1.3.1 Describe the features of C++ language.

1.3.2 Develop C++ program using Integrated Development Environment (IDE).
   - a. Get started with IDE
   - b. Create a project file
   - c. Create a simple C++ program
   - d. Compile a C++ program
   - e. Run a C++ program

### 1.4 Identify the compiling and debugging process, and errors in programming.

1.4.1 Describe the compiling process of a program:
   - a. Source code
   - b. Compiler
   - c. Linker
   - d. Executable file.

1.4.2 Describe with examples the errors in programming:
   - a. Syntax / Compile time errors.
   - b. Run time/ Linker errors.
   - c. Logical errors.

1.4.3 Identify the effective debugging process.

1.4.4 Debug simple programs to demonstrate syntax/ compile time, run time and logical error.
### 2.0 BASIC PROGRAM ELEMENTS

**2.1 Declare variables**
- **2.1.1 Declare variables and constants with an appropriate data types.**
- **2.1.2 Initialize variables.**
- **2.1.3 Identify the problem of uninitialized variables.**
- **2.1.4 Design, implement, test and debug program using data types.**
- **2.1.5 Determine identifier scope: local, global.**
- **2.1.6 Explain keywords.**
- **2.1.7 Design, implement, test and debug program using variables, constants and keywords.**

**2.2 Use input output statements**
- **2.2.1 Identify the syntax used for input and output.**
- **2.2.2 Code programs based on the algorithm that use the input and output statements. Run test and debug the program.**

**2.3 Apply operators and expression**
- **2.3.1 Define operator.**
- **2.3.2 Explain the types of operators:**
  - a. Assignment operators
  - b. Arithmetic operators
  - c. Increment and decrement operators (Unary)
  - d. Relational operators
  - e. Logical operators
  - f. Conditional operators
- **2.3.3 Identify the syntax for each operator with example.**
- **2.3.4 Write expression using operators. Explain typecasting.**
- **2.3.5 Describe operators’ precedence.**
- **2.3.6 Evaluate an expression to identify the operators’ precedence.**
- **2.3.7 Solve problems that apply operators and expression in algorithm and code in a program.**

### 3.0 PROGRAM CONTROL STRUCTURES

**3.1 Understand program control structures**
- **3.1.1 Describe control structure.**
- **3.1.2 Describe the approaches can be chosen depending on problem statement:**
  - a. Sequential
  - b. Selection
  - c. Iterational (Repetition)

**3.2 Solve problems using selection control structures**
- **3.2.1 Identify the selection control structure statement:**
  - a. ‘if’ statement
  - b. ‘switch’ statement
- **3.2.2 Explain the ‘if’ statement type:**
  - a. Simple if statement
  - b. else statement
  - c. else if statement
- **3.2.3 Write an algorithm for each ‘if’ statement type and code into a program based on a problem statement.**
- **3.2.4 Differentiate the assignment (=) and equality (==) operator.**
3.2.5 Apply the nested if.
3.2.6 Describe the switch case statement.
3.2.7 Identify the syntax for switch case statement.
3.2.8 Describe the uses of break statement.
3.2.9 Convert the nested if statement to switch case statement and vice versa.
3.2.10 Solve a given problem by writing algorithm, convert into a program, run, test and debug using selection statement.

3.3 Solve problems using loops control structures
3.3.1 Describe the iterational (repetitive) control structures.
3.3.2 Identify the structure of loops statement:
   a. while
   b. do while
   c. for

3.3.3 Summarize the difference between ‘while’ and ‘do while’ loops.
3.3.4 Describe the advantage of for loops.
3.3.5 Identify the components and structure of the for loops:
   a. Initialization
   b. Termination- condition
   c. Increment- Step

3.3.6 Apply the control transfer statements.
   a. Quitting the loops – break statement.
   b. Continuing the loops- continue statement
   c. Terminating the program using exit() function.

3.3.7 Compare the break, continue and ‘exit’ statement.
3.3.8 Solve a given problem by writing program, run, test and debug using specific loops structure.

4.0 ARRAY, STRUCTURES AND POINTER

4.1 Demonstrate the use of arrays
4.1.1 Define an array.
4.1.2 Identify the characteristics of an array.
4.1.3 Identify when arrays are used based.
4.1.4 Describe the components of an array: index, element and size.
4.1.5 Explain types of array:
   a. one dimensional
   b. two dimensional

4.1.6 Declare one dimensional.
4.1.7 Initialize one dimensional array.
4.1.8 Illustrate the one dimensional array.
4.1.9 Access individual element of one dimensional array.
4.1.10 Declare two dimensional.
4.1.11 Initialize two dimensional array.
4.1.12 Illustrate the two dimensional array.
4.1.13 Access individual elements of two dimensional arrays.
4.1.14 Manipulate array elements.
4.1.15 Solve a given problem by writing program, run, test and debug using array.
4.2 Design program using the concept of pointer
4.2.1 Define pointer.
4.2.2 Identify the syntax to declare a pointer.
4.2.3 Assign the address of variable to pointer.
4.2.4 Manipulate the value of variables using pointer.
4.2.5 Apply new and delete operators.
4.2.6 Design, write, run, test and debug program using pointer.

4.3 Describe the relationship between pointer and array
4.3.1 Identify the relationship between pointer and array.
4.3.2 Apply pointer to manipulate the elements of array.
4.3.3 Solve a given problem by writing program, running, testing and debugging using pointer and array.

4.4 Demonstrate the use of structures
4.4.1 Describe structures.
4.4.2 Identify the difference between structure and arrays.
4.4.3 Define and declare a structure.
4.4.4 Assign values to a structure variable.
4.4.5 Access member variables of a structure.
4.4.6 Apply the coding standards and best practices for structures.
4.4.7 Illustrate structures in memory.
4.4.8 Solve a given problem by writing program, run, and test and debug using structure.
4.4.9 Access member variables of a structure using pointer.

5.0 FUNCTIONS

5.1 Apply the functions in programming
5.1.1 Define function.
5.1.2 Discuss the advantages of functions.
5.1.3 Identify functions based on the problem statements.
5.1.4 Differentiate the library functions and user-defined function.
5.1.5 Apply the coding standards for writing functions.

5.2 Declare function prototypes
5.2.1 Describe the function prototypes
5.2.2 Identify the item in declaring function prototypes:
   a. Function name
   b. Parameters (Data type and Number of values to be passed)
   c. Return type.

5.2.3 Identify the parts of functions:
   a. Function prototype
   b. Function header
   c. Function body

5.2.4 Write a function that:
   a. Accept arguments and return value.
   b. Accept arguments and does not return value.
   c. Do not accept any argument and do not return a value.
5.2.5 Write a function that call user defined functions:
   a. Call function with the constant value.
   b. Call function with variables as arguments.
   c. Call function which does not return any value.
   d. Call function with no arguments and does not return any value.

5.2.6 Identify parameters and arguments.
5.2.7 Apply the best practices for writing functions.

5.3 Identify the scope of variables
5.3.1 Describe the scope of variables.
5.3.2 Explain the local and global variables.
5.3.3 List the differences between local and global variables based on the scope and value after declaration.

5.4 Use the parameters passing techniques
5.4.1 Describe two ways by which function can receive parameters:
   a. Pass by value
   b. Pass by reference

5.4.2 Write a function based on both techniques.
5.4.3 List the difference between ‘Pass by value’ and ‘Pass by Reference’.
5.4.4 Explain the passing arrays as arguments to functions.
5.4.5 Identify the way to call a function to pass an array as argument.
5.4.6 Pass structures to functions.
5.4.7 Write a function to pass an array as argument.

5.5 Understand recursive functions
5.5.1 Describe the concept of recursion and give examples of its use.
5.5.2 Trace the execution of a simple recursive function.
5.5.3 Solve a given problem by writing program, run, test and debug using functions.
### MATRIX OF COURSE LEARNING OUTCOME CONSTRUCTIVE ALIGNMENT

<table>
<thead>
<tr>
<th>Course Learning Outcomes (CLO)</th>
<th>Compliance to PLO</th>
<th>Recommended Delivery Methods</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Explain the fundamental programming constructs element (control structures, arrays, structures, functions and pointers) and articulate how they are used to develop a program. (C2, PLO1)</td>
<td>√ LD1</td>
<td>Interactive Lecture, Discussion and Presentation</td>
<td>Quiz and Test</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td></td>
<td>Final Examination</td>
</tr>
<tr>
<td>2. Apply programming constructs to realise a computer program with debugging techniques to achieve a working program. (C3, P3, PLO1, PLO2)</td>
<td>√ C3</td>
<td>Interactive Lecture, Discussion, Laboratory Activity, Case Study and Presentation</td>
<td>Test and Laboratory Exercises</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Solve computing problems using suitable algorithmic solutions and code these algorithmic solutions in a computer programming language. (C4, P3, A2, PLO1, PLO2, PLO4)</td>
<td>√ C4</td>
<td>Interactive Lecture, Laboratory Activity, Case Study, Discussion and Presentation</td>
<td>Laboratory Exercises and **Project</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remark:**
- LD 1 Knowledge
- LD 2 Practical Skills
- LD 3 Communication Skills
- LD 4 Critical Thinking and Problem Solving Skills
- LD 5 Social Skills and Responsibilities
- LD 6 Continuous Learning and Information Management Skills
- LD 7 Management and Entrepreneurial Skills
- LD 8 Professionalism, Ethics and Moral
- LD 9 Leadership and Teamwork Skills

**Method (s) to assessed the generic skills**
22 ASSESSMENT
The course assessment comprises two components namely:

a. Coursework Assessment (CA) – 50%
Coursework assessments that measure knowledge, practical skills and generic skills are carried out in the form of continuous assessment. Coursework assessments total score comprises the knowledge and practical marks ONLY. It does not include the mark of generic skills.

b. Final Examination (FE) – 50%
Final examination is carried out at the end of the lesson/ instructional session.

ASSESSMENT SPECIFICATION TABLE (AST)

<table>
<thead>
<tr>
<th>COURSE LEARNING OUTCOMES (CLO)</th>
<th>TOPICS</th>
<th>ASSESSMENT METHODS FOR</th>
<th>FINAL EXAMINATION (FE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>COURSEWORK ASSESSMENT (CA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quiz</td>
<td>Test</td>
</tr>
<tr>
<td>1. Explain the fundamental programming constructs element (control structures, arrays, structures, functions and pointers) and articulate how they are used to develop a program. (C2, PLO1)</td>
<td>● ●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Apply programming constructs to realise a computer program with debugging techniques to achieve a working program. (C3, PLO1, PLO2)</td>
<td>● ● ● ●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Solve computing problems using suitable algorithmic solutions and code these algorithmic solutions in a computer programming language. (C4, P3, A2, PLO1, PLO2, PLO4)</td>
<td>● ● ● ●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remark
- Topic 1: Introduction to Fundamentals of Programming
- Topic 2: Basic Program Elements
- Topic 3: Program Control Structures
- Topic 4: Array, Structures and Pointers
- Topic 5: Functions

√ Refers to the CLO to be assessed through the indicated assessment task.
*(#) # refers to the quantity of assessment
● Indicates the topic (s) to be covered under the assigned/ identified assessment tasks. For merged topics, lecturers have the options of choosing the preferred topic (s).
** The generic skills are to be assessed separately. The total score for generic skills is 100%. However, it is NOT PART of the coursework assessment mark.
## DISTRIBUTION OF STUDENT LEARNING TIME (SLT) ACCORDING TO COURSE INSTRUCTIONAL ACTIVITY

<table>
<thead>
<tr>
<th>No.</th>
<th>Learning and Teaching Activity</th>
<th>SLT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEPENDENT LEARNING</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td><strong>Delivery Method</strong></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Lecture</td>
<td>12</td>
</tr>
<tr>
<td>1.2</td>
<td>Practical</td>
<td>48</td>
</tr>
<tr>
<td>1.3</td>
<td>Tutorial</td>
<td>0</td>
</tr>
<tr>
<td>2.0</td>
<td><strong>Coursework Assessment (CA)</strong></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Lecture-hour-assessment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>[1]</td>
</tr>
<tr>
<td></td>
<td>Quiz</td>
<td>[3]</td>
</tr>
<tr>
<td>2.2</td>
<td>Practical-hour-assessment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Laboratory Task</td>
<td>[4]</td>
</tr>
<tr>
<td>2.3</td>
<td>Tutorial-hour-assessment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>[0]</td>
</tr>
<tr>
<td><strong>INDEPENDENT LEARNING</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td><strong>Coursework Assessment (CA)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Projet</td>
<td>[1]</td>
</tr>
<tr>
<td>4.0</td>
<td><strong>Preparation and Review</strong></td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Lecture</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation before theory class eg: download lesson notes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Review after theory class eg: additional references, group discussion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation for quizzes.</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Practical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation before practical class/field work/survey eg: review notes, checklist/ labsheets and/or tools and equipment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Post practical activity eg: lab report, additional references and discussion session</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation before studio work presentation/critique.</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>Tutorial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation for tutorial</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Assessment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preparation for theory test</td>
<td>[0]</td>
</tr>
<tr>
<td></td>
<td>Preparation for final examination</td>
<td>[1]</td>
</tr>
<tr>
<td></td>
<td>Preparation for final assessment</td>
<td>[0]</td>
</tr>
<tr>
<td><strong>FINAL ASSESSMENT (FA)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>[0]</td>
</tr>
<tr>
<td><strong>FINAL EXAMINATION</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Credit = SLT/40</td>
<td>3</td>
</tr>
</tbody>
</table>

**Remark:**

1. Suggested time for
   - Quiz: 10 - 15 minutes
   - Test: 45 - 60 minutes (Theory)
2. 40 hours is equivalent to 1 credit