Course Title: **Digital Logics**
Course No. : ICT Ed. 425
Level: B.Ed.
Semester: Second

Nature of course: Theoretical + Practical
Credit Hour: 3 (2T+1P)
Teaching Hour: 80 (32+48)

1. Course Description

This course provides students with the basic concepts of digital logic, organization and architecture of digital computers as foundation for more advanced computer related studies. It also intends to provides the skill on able to design simple digital devices and implement them. It covers the knowledge area of digital system, logic gates and Boolean algebra, combinational and sequential circuit design, registers, counters, memories and programmable logic devices and VHDL. Laboratory work is essential in this course.

2. Course Objectives

Following are the general objective of this course:

- To make the student knowledgeable about fundamental digital logics and switching networks as well as to exposure of Boolean Algebra and its application for circuit analysis.
- To enable the student to identify the design concept of multilevel gates networks, flip-flops, counters and logic devices.

3. Specific Objectives and Contents

Specific Objectives	Contents
• Differentiate between digital and analog system. • Calculate and converse the number system digital, binary, octal and hexadecimal. • Execute the different number system in arithmetic. • Define codes such as ASCII, EBCDIC & UNICODE. • Explain the error detection and error correction concept.	**Unit 1: Introduction to Digital System (10)** 1.1 Introduction to Analog and digital system 1.2 Features of Digital Systems 1.3. Number Systems- Decimal, Binary, Octal, Hexadecimal and their inter conversions 1.4. Binary Arithmetic. complement system and subtraction using 1's, 2's, 9's, and 10's complement method 1.5. Codes: BCD, XS-3, Gray code, hamming code, alphanumeric codes (ASCII, EBCDIC, UNICODE), 1.6. Error detecting and error correcting codes. ***Lab Work:*** • Practices on Number conversion between Decimal, Binary, Octal, Hexadecimal. • Binary Arithmetic 1's, 2's, 9's, and 10's
• Explain Boolean Logic and Boolean Algebra • Generate the logic gates with diagram, truth table and Boolean function. • Explain Boolean Algebra and laws of Boolean Algebra • Identify the universal gate.	**Unit 2: Logic Gates and Boolean Algebra (10)** 2.1. Basic definition of Boolean Algebra 2.2. Basic Theory of Boolean Algebra, Boolean Functions, Logical operations 2.3. Logic Gates, IC Digital Logic Families. Basic gates (AND, OR, NOT gates) 2.4. Universal gates (NAND and NOR gates), other gates (XOR, XNOR gates) 2.5. Boolean identities, De Morgan Laws.

	Lab Work: • Verification of AND, OR, NOT, NAND, NOR, XOR, and XNOR gate.
• Simplification of Boolean algebra with the use Boolean rules • Solve the Boolean expressions using Boolean algebra, K-Map and Quine McClusky Method	**Unit 3: Simplification of Boolean Functions (10)** 3.1 Simplification of Boolean algebra using Boolean rules 3.2 K-map method (two, three, and four Variable Maps), Don't care conditions 3.3 Canonical and standard forms, product of Sums, sum of product simplification 3.4 NAND and NOR implementation 3.5 Quine McClusky method. ***Lab Work:*** • Apply to simplification Boolean expression using Boolean rules. • Apply to simplification Boolean expression using K-Map Method. • Apply to simplification Boolean expression using Quine McClusky Method.
• Explain combinational circuits • Implement the adder, multiplexers and de-multiplexers • Implement the encoders and decoder • Design combinational circuit design • Design binary and decimal adder	**Unit 4: Combinational Circuit Design (12)** 4.1 Half adder, full adder, half subtracter, and full subtracter. 4.2 Code converters 4.3 Multiplexers and demultiplexers 4.4 Encoders and decoders 4.5 Combinational Circuit design procedure 4.6 Binary Parallel Adder 4.7 Decimal Adder ***Lab Work:*** • Design the adder and subtracter • Implement logic of Mux/Demux and Encoder/Decoder • Design the number system converter circuit • Design Various Decision making circuits.
• Explain Sequential Logic Circuits • Design Flip-flops • Create flip-flop excitation table	**Unit 5: Sequential Circuit Design (7)** 5.1. Flip-flops: RS, JK, D, and T, Latches 5.2 Triggering of flip-flops 5.3 Master slave flip flop 5.4 Flip-flop excitation table and design procedure. 5.5 state diagram and simple sequential circuits ***Lab Work:*** • Design the different types of flip-flops.

• Explain counters and Shift Registers. • Define electronics part of memories • Describe digital logic families • Analyze and design synchronous sequential circuits • Analyze asynchronous sequential circuits	**Unit 6: Registers, Counters, Memories and Programmable Logic Devices (15)** 6.1 Resisters, Shift registers 6.2 Analysis of synchronous sequential circuit 6.3 Design of synchronous sequential Circuits: Counters, state diagram, state reduction, state assignment 6.4 Analysis of asynchronous sequential circuit 6.5 Problems of asynchronous sequential circuit design 6.6 Memories: ROM, PROM, EPROM 6.7 PLD, PLA 6.8 Digital Logic Families: TTL, ECL, and CMOS ***Lab Work:*** • Design any clock driven sequential circuit • Design verify the principle of conversion of parallel data into serial.
• Define concept of VHDL • Design simple circuits by using VHDL	**Unit 7: VHDL (10)** 7.1 RTL Design with VHDL 7.1.1 Shape of VHDL 7.1.2 Data Types 7.1.3 Concurrent Statements 7.1.4 Processes and Variables 7.1.5 Simulating a Simple Design 7.1.6 Creating Memory 7.1.7 Finite State Machines 7.1.8 Loops and Conditional Elaboration 7.1.9 Attributes 7.1.10 Functions and Procedures **LAB Work:** • Demonstrate the different circuit in VHDL Tools
• Design real world logic circuits using VHDL or any other hardware design tools	**Unit 8. Project Work (6)**

4. Instructional Techniques

The instructional techniques for this course are divided into two groups. First group consists of general instructional techniques applicable to most of the units. The second group consists of specific instructional techniques applicable to particular units.

4.1 General Instructional Techniques

Reading materials will be provided to students in each unit. Lecture, Discussion, use of multi-media projector, brain storming are used in all units.

4.2 Specific Instructional Techniques

Demonstration is an essential instructional technique for all units in this course during teaching learning process. Unit one and three are theoretical and numerical chapters so, they require more exercise and demonstration of principles. Use more pictures, flowchart of method, and assignment. Specifically,

demonstration with practical works will be specific instructional technique in this course. The details of suggested instructional techniques are presented below:

Units	Activities
Unit 2: Logic Gates and Boolean Algebra	• Verify AND, OR, NOT, NAND, NOR, XOR, and XNOR gate using physical real bread board and two input TTL ICs. • Demonstration by the teacher on physical real device and circuitry design to demonstrate the working principle, objective and their use. • Individual lab work of real bread board by each student • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 4: Combinational Circuit Design	• Design the adder and subtracter, Implement logic of Mux/Demux and Encoder/Decoder, and Design the number system converter circuit, Design Various Decision making circuits. • Lab work in pairs in different tasks assigned by the teacher • Monitoring of students' work by reaching each pair and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 5: Sequential Circuit Design	• Design RS, JD, D and T flip-flops with their excitation table and design procedure. • Demonstrate the objective, use and practically implement the master slave flip-flop. • Demonstrate the state diagram of any simple sequential circuit. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 6: Registers, Counters, Memories and Programmable Logic Devices	• Demonstrate the implement the concept, objective and real use of registers, counters, memories and PLDs. • Design any clock driven sequential circuit, verify the principle of conversion of parallel data into serial. Design circuits like: digital clock, voting system, counting machine, storage device, traffic control system, frequency division circuits, and analyze circuits. • Demonstrate the interfacing method with various types of logic families and integrated circuits. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 7: VHDL	• Explain the fundamental of VHDL programming language using class lecture method. • VHDL language should be taught to specify the logic circuits. Instructor should illustrate how VHDL can be used to specify the desired functionality and how CAD tools provide a mechanism for developing

	the required circuits. Instructor should assign design projects like Adder/Subtracter, Multiplexer/Demultiplexer, Encoder/Decoder, Flip-flops, resister and counters to each individual using both methodologies: manual design and CAD tools to design logic circuits.
Project Work	• Design any real world digital logic circuit using combinational and sequential circuits. • Use VHDL programming language. • It is a project to be completed by individual student under the direct supervision of project supervisor.

5. Evaluation :

Internal Assessment	External Practical Exam/Viva	Semester Examination	Total Marks
40 Points	20 Points	40 Points	100 Points

***Note**: Students must pass separately in internal assessment, external practical exam and semester examination.*

5.1 Internal Evaluation (40 Points):

Internal evaluation will be conducted by subject teacher based on following criteria:

1) Class Attendance 5 points
2) Learning activities and class performance 5 points
3) First assignment (written assignment) 10 points
4) Second assignment (Case Study/project work with presentation) 10 points
5) Terminals Examination 10 Points

Total 40 points

5.2 Semester Examination (40 Points)

Examination Division, Dean office will conduct final examination at the end of semester.

1) Objective question (Multiple choice 10 questions x 1mark) 10 Points
2) Subjective answer questions (6 questions x 5 marks) 30 Points

Total points 40

5.3 External Practical Exam/Viva (20 Points):

Examination Division, Dean Office will conduct final practical examination at the end of semester.

6. **Recommended Books and References materials**

Recommended Books:

Floyd, T. L. (2009). *Digital fundamentals* (10th ed). Upper Saddle River, N.J: Pearson/Prentice Hall.

Mano, M. M., & Kime, C. R. (2008). *Logic and computer design fundamentals* (4. ed). Upper Saddle River, NJ: Pearson Prentice Hall.

References materials:

Brown, S. D., & Vranesic, Z. G. (2014). *Fundamentals of digital logic with Verilog design* (Third edition). New York: McGraw-Hill Higher Education.

Rafiquzzaman, M. (2005). *Fundamentals of digital logic and microcomputer design* (5th ed). Hoboken, N.J: J. Wiley & Sons.

Mano, M. M. (2002). *Digital design* (3rd ed). Upper Saddle River, NJ: Prentice-Hall.

Course Title: **Object Oriented Programming with C++**
Course No. : ICT. Ed. 426 Nature of course: Theoretical + Practical
Level: B.Ed. Credit Hour: 3 hours (2T+1P)
Semester: Second Teaching Hour: 80hours (32+48)

---

1. Course Description

The aim of the course is to develop the skill on thinking about computation and problem solving in Object Oriented Paradigms. The course helps the students to discover the basic concepts of object-oriented programming concept such as object, class, inheritance, polymorphism, abstraction and encapsulation and apply in C++. Students are more engaged in laboratory work to exaction of programing experiments rather than theoretical concept.

2. General Objectives

Following are the general objective of this course:

- To acquaint the student with fundamentals object oriented paradigms and programming style in C++ programming language.
- To develop the skill on apply object oriented programming concept in programming.
- To enable a student in explore the new software development paradigms.

3. Course Outlines:

Specific Objectives	Contents

• Compare procedure and object oriented programming concept • Describe the feature of object oriented programming. • List out the C++ compilers • Compare coding structure of C and C++. • Demonstrate the C++ programming styles.	**Unit 1: Concept of Object Oriented Programming (12)** 1.1 Programming Languages and Software Crisis 1.2 Procedure Vs Object Oriented Programming Language 1.3 Feature of Object Oriented Programming 1.4 Popular Object Oriented Programming Language and features 1.5 Advantage and Disadvantage of OOPs 1.6 Introduction of C++ and Compilers 1.7 Programming Structure in C++ 1.8 Comparison on C and C++ 1.9 Additional Data types, token in C++ 1.10 Insertion and Extraction Operators **Practical Works:** • Install the compiler of C++. • Use Insertion and Extraction Operator. • Compare the C and C++ Compiler and structure
• Explain the Object and Class • Define Data member and Member function. • Define inline member function. • Use array in member function and objects. • Define static and friends function. • Explain constructor and destructors.	**Unit 2: Object and Class (16)** 2.1 Concept of Object and Class 2.2 Define Data Member and Member Function 2.3 Create object and access Member Function 2.4 Making outer function inline 2.5 Array with in Class 2.6 Array of Objects 2.7 Static Data Member and Static Function 2.8 Friends Functions 2.9 Concept of Constructor and Destructor 2.10 Empty, Parameterized and Copy constructor 2.11 Define Destructor **Practical Works:** • Create class and objects with data member and member function. • Declare and define member function and data member with visibility. • Create static function • Create friend functions. • Create different types of constructors
• Explore the concept of constructor and Destructors. • Apply Binary operator and unary operator overloading.	**Unit 3: Operator Overloading (12)** 3.1 Concept of Operator Overloading 3.2 Defining Operator Overloading 3.3 Rules of Operating Overloading 3.4 Unary Operator Overloading 3.5 Return types in overloading function

• Describe data conversion methods.	3.6 Binary Operator Overloading 3.7 Manipulation String using Operator Overloading 3.8 New and Delete Operator Overloading 3.9 Data Conversion **Practical Works:** • Create unary operator overloading. • Apply different types of operator overloading function return methods. • Apply binary operator overloading. • Create Data conversion methods
• Explore the concept of inheritance • Describe the base class and access specifier . • Apply single, multiple, multilevel inheritance. • Use constructor in Derived class.	**Unit 4: Inheritance (12)** 4.1 Concept of Inheritance 4.2 Base and Derived Class 4.3 Private, Public and Protected Specifier 4.4 Derived class declaration 4.5 Member function overriding 4.6 Single, Multiple, multilevel and hybrid Inheritance 4.7 Ambiguity problems in inheritance 4.8 Constructor in Derived Class 4.9 Extending operator overloading in derived class **Practical Works:** • Create single level inheritance. • Create multiple inheritance. • Create multilevel inheritance. • Check the ambiguity problems.
• Revision concept of pointer. • Identify need of virtual function. • Describe Virtual function. • Describe the Pure virtual function. • Describe the Abstract and container class	**Unit 5: Virtual Function and Polymorphism** (8) 5.1 Concept of Pointer 5.2 Need of virtual function 5.3 Definition of Virtual Function 5.4 Pure Virtual function 5.5 Abstract Class 5.6 Container class **Practical Works:** • Create virtual function. • Create pure virtual function. • Create Abstract and container class.

• Explain concept of template. • Define function template and class template. • Apply error handling in programming. • Apply the different exception handling methods.	**Unit 6: Template and Exception Handling (8)** 6.1 Concept of Template 6.2 Function overloading and problems 6.3 Function Template 6.4 Overloading function template 6.5 Class Template 6.6 Derived class template 6.7 Concept of error handling 6.8 Basic of exception handling 6.9 Exception handling mechanism: throw, catch and try **Practical Works:** • Create and apply function template. • Create and apply template class. • Apply try, catch and throw methods in program.
• Describe the concept the procedure oriented paradigms. • Describe Object oriented paradigms. • Analysis complexity in software development. • Describe object oriented analysis and design methods.	**Unit 7: Object Oriented System Development (6)** 7.1 Procedure oriented paradigms 7.2 Procedure oriented development Tools 7.3 Object Oriented Paradigms 7.4 Object-Oriented Programming as a New Paradigm 7.5 Computation as Simulation 7.6 Coping with Complexity's 7.7 Reusable Software 7.8 Object-Oriented analysis and Design **Practical Works:** Case study on comparison of procedure and object oriented paradigms. .
• Create console application using C++.	**Unit 8: Project (6)** Develop simple Application using (6) C++ with the feature of class, object, inheritance, polymorphism and encapsulation.

4. Instructional Techniques

The instructional techniques for this course are divided into two groups. First group consists of general instructional techniques applicable to most of the units. The second group consists of specific instructional techniques applicable to particular units.

4.1 General Techniques

Reading materials will be provided to students in each unit. Lecture, Discussion, use of multi-media projector, brain storming are used in all units.

4.2 Specific Instructional Techniques

Demonstration is an essential instructional technique for all units in this course during teaching learning process. Specifically, demonstration with practical works will be specific instructional technique in this course. The details of suggested instructional techniques are presented below:

Units	Activities
Unit 1: Concept of Object Oriented Programming	• Select and Install the different compiler of C++. • Demonstrate the programming structure of C++. • Compare the other program provide the assignment for understanding of objects oriented paradigms. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 2: Object and Class	• Demonstrate class and object creation methods in C++. • Demonstrate the methods and attributes in Class and access from objects. • Demonstrate the different types of methods such as inline, statics and friends. • Lab work in pairs in different tasks assigned by the teacher • Monitoring of students' work by reaching each pair and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 3: Operator Overloading	• Demonstrate the unary and binary operator overloading methods. • Lab work in pairs in different tasks assigned by the teacher • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 4: Inheritance	• Demonstrate the single, multiple and multilevel inheritance and applied into C++. • Lab work in pairs in different tasks assigned by the teacher. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 5: Virtual Function and Polymorphism	• Demonstrate the virtual and pure virtual functions and application. • Demonstrate the abstract and container class. • Lab work in pairs in different tasks assigned by the teacher. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback

Unit 6: Template and Exception Handling	• Demonstrate the template function and class. • Demonstrate the exception handling concept in OOPs with reference C++. • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 8: Project	• Develop console application applied with OOPs Concepts.

7. Evaluation :

Internal Assessment	External Practical Exam/Viva	Semester Examination	Total Marks
40 Points	20 Points	40 Points	100 Points

***Note**: Students must pass separately in internal assessment, external practical exam and semester examination.*

7.1 Internal Evaluation (40 Points):

Internal evaluation will be conducted by subject teacher based on following criteria:

6) Class Attendance 5 points
7) Learning activities and class performance 5 points
8) First assignment (written assignment) 10 points
9) Second assignment (Case Study/project work with presentation) 10 points
10) Terminal Examination 10 Points

Total 40 points

7.2 Semester Examination (40 Points)

Examination Division, Dean office will conduct final examination at the end of semester.

3) Objective question (Multiple choice 10 questions x 1mark) 10 Points
4) Subjective answer questions (6 questions x 5 marks) 30 Points

Total points 40

7.3 External Practical Exam/Viva (20 Points):

Examination Division, Dean Office will conduct final practical examination at the end of semester.

5. Recommended books and References materials (including relevant published articles in national and international journals)

Recommended books:

Balagurusamy, E. (2013). *Object oriented programming with C++*. New Delhi: Tata McGraw-Hill (Unit 1-8).

BaralDayasar&BaralDiwakar(2010), Secrete of Object Orientd Programming in C++, Kathmandu, BhundipuranPrakashan (Unit 1-8).

References materials:

Robert Lafore(2003), Object Oriented Programming in Turbo C++, Galgotia Publications Ltd. India, 2003 (Unit 1-8).

Schildt, H. (2003). *C++: the complete reference* (4th ed). New York: McGraw-Hill.

Lippman, S.B., Lajoie. J., C++ Primer, 3rd Ed., Addison Wesley, 1998