MATHEMATICAL FOUNDATIONS OF COMPUTER SCIENCE

Course Objectives:

1. To introduce the students to the topics and techniques of discrete methods and combinatorial reasoning.
2. To introduce a wide variety of applications. The algorithmic approach to the solution of problems is fundamental in discrete mathematics, and this approach Reinforces the close ties between this discipline and the area of computer science.

Course Outcomes:

1. This course enables students of computer science to develop applications in areas of data structures, the theory of computer languages, and analysis of algorithms.

Prerequisites: Sets, Mathematical Notations

PART A

Fundamentals of logic : Basic connectives and truth tables, logic equivalence, the laws of logic, logical implications-rules of inference. The use of quantifiers, quantifiers, definitions and the proof of theorems.

Counting : Permutations, Combination, Pigeonhole Principle, Elements of Probability, Recurrence Relations. 06 Hrs

Relations and Digraphs: Product Sets and Partitions, Relation and Digraphs, Paths in Relations and Digraphs, Properties of Relations and Digraphs, Properties of Relations, Equivalence Relations, Datastrutures for Relations and Digraphs, Operations on Relations, Tansitive Closure and Warshall’s Algorithm. 07 Hrs

Functions: Function, Function for Computer Science, Growth of functions, Permutation functions, 06 Hrs

Order Relations and Structure: Partially Ordered Sets, External Elements of Partially, Ordered Sets, Lattices, Finite Boolean Algebras, Functions on Boolean Algebras, Circuit Design. 07 Hrs

PART B

Trees: Trees, Labeled Trees, Trees Searching, Undirected Trees. Minimal Spanning Trees. 06 Hrs
**SemiGroups And Groups:** Binary Operation Revisited, Semigroups, Product and Quotillents of Semigroups, Groups, Product and Quotients of Groups, Cyclic groups, Cosets and Lagrange’s theorem.  
08 Hrs

**Languages and Finite-State Machines:** Languages, Representations of Special Grammers and Languages, Finite-State Machines, Monoids, Machines and Languages.  
06 Hrs

**Coding theory:** Elements of coding theory, Hamming metric, Parity check and generator matrices, Group codes: Decoding with coset leaders, Hamming matrices.  
06 Hrs

**Text Books:**


**References.**


---

**COMPUTER ORGANIZATION**

<table>
<thead>
<tr>
<th>Subject Code : 11CS33</th>
<th>Credits :03</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.I.E : 50 Marks</td>
<td>S.E.E : 50 Marks</td>
</tr>
<tr>
<td>Hours/Week : 3hrs (Theory)</td>
<td>Total hrs :42</td>
</tr>
</tbody>
</table>

**Course Objective:**

1. The computer organization is concerned with the structure and behavior of digital computers. The main objective of this subject to understand the overall basic computer hardware structure, including the peripheral devices
2. The aim of the subject is to provide a through discussion of the fundamentals of computer organization and architecture and to relate these to contemporary design issues.
Course Outcomes:
1. Helps to identify major hardware components.
2. Design alternate methodologies and assess cost/ performance trade off while assessing a computing system.
3. Write an assembly language program.

Prerequisites: Logic Design, Programming language.

PART A

Basic Structures of Computers, machine instructions & programs
Basic Operational Concepts: Bus Structures, Performance, Processor clock, Basic Performance equation, Pipelining & Superscalar operation, Clock rate, Performance measurement, Multiprocessor & Multicomputer, Evolution of Performance. Numbers, Arithmetic operations and characters, Memory Locations & Addresses, Byte addressability, Big-endian & Little-endian assignments, Word Alignment, Accessing Numbers, Characters & Character strings, Memory Operation, Instruction & Instruction Sequencing, Register Transfer Notation, Assembly Language Notation. Basic Instructions types. 06 Hrs

Machine Instructions & Programs: Instruction Execution & Straight-line sequencing, Branching, Condition Codes, Generating Memory Addresses, Address Modes: Implementation of Variables & Constants, Indirection & Pointers, Indexing & Arrays, Relative Addressing, Additional Modes. Assembly Language: Assembler Directives, Number Notation, Basic Input/Output Operations, Stacks & Queues: Subroutines, Subroutine Nesting & Processor Stack, Parameter Passing, The Stack Frame, Additional Instructions, Logic Instruction, Shift & Rotate Instructions, Multiplication & Division, Encoding of Machine Instruction. 05 Hrs

Input/Output Organization:
Accessing I/O devices: Interrupts, Interrupt Hardware, Enabling & Disabling Interrupt, Handling Multiple devices, Controlling Device Requests, Exceptions. 05 Hrs

Direct Memory Access:
Bus Arbitration, Buses: Synchronous Bus, Asynchronous Bus, Interface Circuits: Parallel Port, Serial Port, Standard I/O interfaces, PCI bus, SCSI bus, USB Bus. 05 Hrs

PART B

The Memory System & Some Basic Concepts:
Semiconductor RAM Memories: Internal Organization of Memory Chips, Static Memories, Asynchronous DRAMs, Synchronous DRAMs, Structure of Larger Memories, Memory System Considerations, Read-only Memories: ROM, PROM, EPROM, EEPROM, Flash memory, Speed, Size & Cost. Cache Memories: Mapping functions. Performance considerations: Interleaving, Hit Rate & Miss Penalty. Virtual memories: address translation Memory management requirements. Secondary Storage. 06 Hrs

Arithmetic Addition & Subtraction of Signed Numbers:
Addition/Subtraction Logic Unit, Design of fast adders: Carry-Look ahead Addition, Multiplication of Positive numbers: Signed-Operand Multiplication: Booth Algorithm, Fast

**Basic Processing Unit:**

**Some Fundamental Concepts:** Register Transfers, Performing an Arithmetic or Logic operation, Fetching a Word from Memory, Storing a Word in Memory. Execution of a Complete Instruction: Branch instruction. Performance Consideration: Multiple-Bus Organization. Hardwired Control: A Complete Processor, Micro programmed Control.  

**Pipelining**  
**Basic concepts:** role of cache memory, pipelining performance  

**Data hazards:** operand forwarding, handling data hazards in software some side effects.  

Instruction hazards: Unconditional branches, conditional branches and branch prediction.  

**Text Book:**  

**Reference Books:**  

**DATA STRUCTURES WITH C**

<table>
<thead>
<tr>
<th>Subject code: 11CS34</th>
<th>Credits : 04</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.I.E : 50 Marks</td>
<td>S.E.E : 50 Marks</td>
</tr>
<tr>
<td>Hours/Week : 4 hrs (Theory)</td>
<td>Total Hours : 52</td>
</tr>
</tbody>
</table>

**Course Objective:**

1. Describe the representation of numeric and character data.  
2. Understand how precision and round-off can affect numeric calculation.  
3. Discuss the use of primitive data type and built-in data structure.  
4. Describe common applications for each data structure in the topic list.  

**Course Outcome:**

At the end of this course the student should be able to  

1. Understand select various data structure and algorithms function.  
2. Implement various data structures in a high-level language.  
3. Selection of an appropriate data structure for a given problem.
Prerequisites:
  Knowledge of any high level programming language.

PART A

C language features

Structures and Unions: Structure definition, Giving value to members, Structure initialization, Comparison of structure variables, Arrays of structures, Arrays within structures, Structures within structures, Structure and functions, Unions, Size of structures, Bit-fields.

Pointers: Understanding pointers, & the address of operator, Declaring and initializing pointer, Accessing a variable through it's pointer, Pointer and arrays, Pointer and character strings, Pointer and functions, Pointer and structures. 06 Hrs.

Dynamic memory allocation: Meaning of dynamic memory allocation, MALLOC, CALLOC, Free and REALLOC functions, Pointer revisited.

File management: Definition and opening a file, Closing a file, I/O operations on files, Error handling during file operation, Random access to files, Command line arguments 06 Hrs.

Stack

Definition and examples: primitive operation, Example. Representing stack in C: Implementing the pop( ) operation, Testing for exceptional conditions, Implementing the push( ) operation Example: Infix, Postfix and Prefix, Basic definitions and Examples, Evaluating a postfix expression, Program to evaluate postfix expression, Converting an expression from infix to postfix, Program to convert expression from infix to postfix. 07 Hrs.

Recursion

Recursive definition and processes: Factorial function, Multiplication of natural numbers, Fibonacci sequence, Binary search, Properties of recursive definition or algorithm.

Recursion in C: Factorial of a number, Generation of Fibonacci numbers, Binary searching, Towers of Hanoi problem. 07 Hrs.

PART B

Queues and Lists

The queue and it's sequential representation: C implementation of queues, Insert operation, Priority queues, Array implementation of priority.

Linked lists: Inserting and removing nodes from a list, Linked implementation of stacks, Getnode and Freenode operations, Linked implementation of queues, Linked list as a data structure, Example of list operations, Header nodes. 06 Hrs.

Lists in C: Array implementation of list, Linked implementation of lists, Limitations of array implementation, Allocating and freeing dynamic variables, Linked list using dynamic variable, Queues as lists in C, Example of list operations in C, Non-integer and non-homogeneous lists.

Other list structures: Circular lists, Stacks as circular list, Queues as a circular list, Primitive operations on circular list, Doubly linked list. 07 Hrs.

Trees


Trees and their applications: C representation of trees, Tree traversals, General expression as trees, Evaluating an expression tree, Constructing a tree. 07 Hrs.
**Sorting And Searching**

**06 Hrs.**

**Text Book:**

**Reference Books:**

---

**LOGIC DESIGN**

<table>
<thead>
<tr>
<th>Sub Code</th>
<th>Credit</th>
<th>C.I.E : 50 Marks</th>
<th>S.E.E : 50 Marks</th>
<th>S.E.E :03 hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>11CS35</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hours/Week</td>
<td>3 hrs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total hours</td>
<td></td>
<td></td>
<td></td>
<td>42</td>
</tr>
</tbody>
</table>

**Course Objectives:**
1. Design a simple circuits using fundamental building blocks.
2. Appreciate the effect of AND, OR, NOT operations on binary data.

**Course Outcomes:** At the end of this course student have the
1. Knowledge of designing basic logic Circuits.
2. Knowledge of basic memory elements.
3. Knowledge of designing combinational and sequential circuits.

**Prerequisites:**
Basic Electronics

**PART A**

**Number Systems, operations and conversions:**
Decimal, binary, octal, hex codes, BCD, parity, gray, ASCII, Unicode, Digital logic: overview of basic gates and universal logic gate, positive and negative logic, Introduction to VHDL.  

**05 hrs**
Combinational Logic Circuits:
Boolean laws and theorems, principle of duality. Normal formulas: canonical formulas, minter canonical formulas, m-notations, Max term canonical formulas, M-notation; combinational networks: analysis procedure, synthesis procedure. Incomplete Boolean functions and don’t care conditions: Describing incomplete Boolean functions don’t care conditions in logic design. 05 hrs

Simplification of Boolean Expressions:
Karnaugh Maps: One variable and two variable maps, three variable and four variable maps, karnaugh maps and canonical formulas, Product and sum term representation on k-map. Using karnaugh maps to obtain minimal expression for complete Boolean functions. Minimal expressions of incomplete Boolean functions. The quine-Mccluskey method of generating prime implicants and prime implicates. 06 hrs

Logic Design with MSI components:
Binary adders and sub tractors ,  carry look-ahead adder; decimal adders; comparators; decoders; logic design using decoders; Encoders; multiplexers; logic design with multiplexers. 05 hrs

PART B

Programmable logic devices:
PLD notations; programmable Read only memories, programmable logic arrays; programmable array logic devices, VHDL language for combinational circuit design. 05 Hrs

Sequential circuits:
Flip-flops : RS flip-flop, gated flip-flops, Edge-triggered RS, D, T, JK flip-flop, Flip=flop timings, JK master slave flip-flops, conversion flip flops. 06 Hrs

Sequential circuits:
Registers. Counters, Binary ripple-counters, synchronous binary counters, counters based on shift registers. 05 Hrs

Design of synchronous counters:
Design of mod-6 counters using clocked JK flip-flops. Design of synchronous of mod 6 Counter using clocked D,T, or SR Flip flops. 05 Hrs

Text Books:
Reference Books:

   Tata McGraw-Hill

OBJECT ORIENTED PROGRAMMING

Objectives:
Object orientation is an approach to understand the complexities of the real world. In contrast to the earlier approaches like procedural etc, object orientation helps to formulate the problems in a better way giving high reliability, adaptability and extensibility to the applications. The course emphasis is on the object orientated facilities of C++ and how they can be used to create structured, modular and re-usable code. Object programming also provides a base for the further object oriented programming like Java, C# and .Net.

Course Outcomes: At the end of course student should understand
1. The object oriented programming concepts.
2. Understand the differences between OOPS and conventional structured programming approaches.
3. Student should be able to appreciate the paradigm of OOPs.
4. Develop the skills in coding, debugging and implementation of algorithms using C++.

Prerequisites: High level language C.

PART A

Object-Oriented Paradigm: Introduction to program paradigms, OOP’s , A new paradigm, Evolution of programming paradigms, Structured versus object-oriented development, Elements of object-Oriented programming, Objects, Classes, Multiple views of the same object, Encapsulation and data abstraction, Inheritance, Delegation – Object composition, Polymorphism.

Arrays and Strings: Introduction, Operations on arrays, Array illustrations, Multi-dimensional arrays, Strings, Strings manipulations, Arrays of strings, Evaluation order/Undefined behaviors.

07 Hrs

Modular programming with functions: Introduction, Function components, Passing data to functions, Function return data type, Library Functions, Parameter passing, Return by reference,
Default arguments, Inline functions, Function overloading, Function Templates, Arrays an
functions, C++ Stack, Scope and extent of variables, Storage Classes, Functions with variables
number of arguments.

Classes and Objects: Introduction, Class Specification, Class Objects, Accessing class members,
Defining member functions, Outside member functions as inline, Accessing member functions
within the class, Data hiding, Access boundary of objects revisited, Empty classes, Pointers within
a class, Passing objects as arguments, Returning objects from functions, friend functions and friend
classes, Constant parameters and member functions, Structures and Classes.

Object Initialization: Constructors, Parameterized constructors, Destructor, Constructor
overloading, Order of construction and destruction, Constructors with default arguments, Dynamic
initialization through constructors, Constructors with dynamic operations, Copy constructor,
Constructors for two-dimensional arrays, Constant objects and constructor, Static data members
with constructors and destructors.

PART B

Operator Overloading: Introduction, Overloadable operators, Unary operator overloading,
Operator keyword, Operator return values, Binary operator overloading, Arithmetic operators,
Concatenation of strings, Comparison operators, Arithmetic assignment operators, Overloading of
new and delete operators, Data conversion, Overloading with friend functions, Assignment
operator overloading.

Inheritance and Virtual Functions:

Introduction, Derived class declaration, Forms of inheritance, Inheritance and member
accessibility, Constructors in derived classes, Destrucrtors in Derived classes, Abstract Classes,
Multilevel inheritance, multiple inheritance, Hierarchical inheritance, Multipath inheritance and
virtual base class, Hybrid inheritance.

Virtual Functions: Introduction, Need for virtual functions, Pointer to derived class objects,
definition of virtual functions, Array of pointers to base class objects, pure virtual functions.

Generic Programming with Templates:

Introduction, Function templates, Overloaded Function Templates, Nesting of Function
Calls, Multiple Argument Function Template, Class Templates, Inheritance of class templates,
Class Templates with Overloaded Operators.

Stream Computations with Console-

Introduction, Predefined Console Streams, Hierarchy of Console Stream Classes, Unformatted
Console I/O Operations, Formatted Console I/O Operations., Manipulators, Stream Operators with
User-defined Classes

Stream Computation with Files –Introduction . Hierarchy of File Stream Classes, Opening and
Closing of Files, File Modes File Pointers and their Manipulations.
Text books:


Reference books:


Data Structures with C LAB

<table>
<thead>
<tr>
<th>Subject code: 11CSL37</th>
<th>Credits : 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.I.E : 50 Marks</td>
<td>SEE : 50 Marks</td>
</tr>
<tr>
<td>Hours/Week : 2 hrs (Practical)</td>
<td>S.E.E : 3 hrs</td>
</tr>
</tbody>
</table>

Course Objectives:
At the end of the course the student should understand the basic principles of data structures.

Course Outcomes:
At the end of this course the student should be able to
1. Implement the user-defined data structures in a high-level language.
2. Compare alternative implementations of data structures with respect to performance.
3. Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues and trees.
4. Compare and contrast the costs and benefits of dynamic and static data structure implementations.
5. Choose the appropriate data structure for modeling a given problem.

PART-A

1. Write a program to work on strings, to find out string length and changing the case using pointers and without using library functions.

2. Write a program to calculate average of student-marks for a student database.

3. Write a program to illustrate the method of sending an entire structure to a function.
4. Write a program to calculate Binomial Coefficient using pointers.
5. Write a program to search for an element in a sorted list employing binary search using recursion.

**PART-B**

1. Write a program to perform push, pop operations on stack and display the contents on a stack. Implement stack using linear array.
2. Write a program to perform push, pop operations on stack and display the contents. Implement stack using pointers.
3. Write a program to convert a postfix expression to infix expression and an infix expression to postfix expression.
4. Write a program that evaluates a postfix expression and prints the result.
5. Write a program to perform insert, delete operations on queue and display the contents. Implement queue using linear array.
6. Write a program to perform insert, delete operations on queue and display the contents. Implement queue using pointers.
7. Write a program to create a singly linked list to perform addition, deletion and insertion of newly created node at any position in the list.
8. Write a program to reverse singly linked list.
9. Write a program to accept 2 singly linked lists and create a new singly linked list which contains the elements that are common in both the lists.
10. Write a program to create a doubly linked list to perform addition, deletion and insertion of newly created node at any position in the list.
11. Write a program to concatenate two singly linked lists.
12. Write a program to add two long positive integers using circular linked lists.
13. Write a program to create, add, remove and display elements from circular linked list.
14. Write a program to implement stack using lists.
15. Write a program to implement queue using lists.
16. Write a program to create a tree and perform inorder, preorder and postorder tree traversal techniques.
17. Write a program to construct a binary search tree and count the number of leaves in a binary tree.

LOGIC DESIGN LAB

<table>
<thead>
<tr>
<th>Subject code: 11CSL38</th>
<th>Credits : 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIE : 50 Marks</td>
<td>SEE : 50 Marks</td>
</tr>
<tr>
<td>Hours/Week : 2 hrs (Practical)</td>
<td>SEE : 3 Hrs</td>
</tr>
</tbody>
</table>

Course Objective:
1. Realization of Boolean expression using combinational circuits.
2. Design Combinational and sequential circuits.
3. Test and verify Boolean expression for different MSI designs like MUX, Decoder.

Course Outcome:
1. At the end of the course student should be able to design independently different application oriented circuits.

Experiments:
1. Simplification, Realization of Boolean Expressions using Combinational Logic circuits.
2. Realization of Half adder/Full adder, Half Subtractor/Full Subtractor Using logic gates.
4. Implementation of BCD to excess-3 and excess-3 to BCD code converters.
5. Design and implementation of BCD to Gray and Gray to BCD converters.
6. Realization of Two bit magnitude comparator using 7485 chip.
8. Verification of Truth tables for different sequential circuits viz JK master slave, T-Flip flop, D Flip flop.
10. Realization of shift right and shift left operations using 7495 IC a 4 bit shift register.
11. To design and implementation of Ring counter/Johnson counter using 7495 IC.
12. Study of 3-bit digital to analog converter using R-2R
OBJECT ORIENTED PROGRAMMING LAB

<table>
<thead>
<tr>
<th>Subject Code: 11CSL39</th>
<th>Credits: 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIE : 50 Marks</td>
<td>SEE : 50 Marks</td>
</tr>
<tr>
<td>Hrs/Week : 2 hrs. (Practical)</td>
<td>S.E.E : 03 hrs.</td>
</tr>
</tbody>
</table>

**Course Objectives:**

i. Hands on experience of Object Oriented Programming Concepts with C++.

ii. Preparedness to study independently any other Object Oriented Programming language and apply to variety of realtime problem scenarios.

**Course Outcomes:** At the end of the course, the student should be able to:

- Conceptualize the given problem and transform it into an Object oriented system,
- Implement coding standard and verification practices and build expertise in Object Oriented programming language.

**PART – A**

1) Write a C++ program to detect whether a given year is leap or not.

2) Write a C++ program to check whether given number is Odd or Even. Using ternary operator.

3) Write a C++ program to count vowels, spaces and consonants from a given sentences using switch.

4) Write a C++ program to display youngest and eldest person, age from a given set of N number of person (array).

5) Write a C++ program to find Addition and Subtraction of two M x N matrices.

6) Write a C++ program to concatenate two strings.

7) Write a C++ program to find factorial of a number using recursive function.

8) Write a C++ program to store numbers into an array and find sum of all the elements of the array using pointers.

9) Write a C++ program to store numbers in two arrays, namely, A and B. Add two numbers, picking from the arrays from a given position in the arrays using pointers.
PART – B

1) Create a simple class STUDENT containing the data members, roll no,name,age &
display the contents using user defined functions. Test the program with and
without scope resolution operator.

2) Create a class called, EMPLOYEE containing data members,Empno,
Empname,Designation,Basicpay,DA,HRA,Insurance,TAX. Develop the member
functions for calculating and displaying the Netsalary
\[(\text{Netsalary}=(\text{Basicpay}+\text{DA}+\text{HRA})-(\text{Insurance}+\text{TAX}))\].

3) Write a C++ program to create class DATE. Display age of the person by considering
date of birth and current date using inline function.

4) Write a C++ program to create a class ACC with data members, accno,balance.
Create objects, namely, ACC1 and ACC2. Write a member function to transfer
amount from ACC1 to ACC2. Display the new balance in the transacted accounts.

5) Create a class FIXED-DEPOSIT with data members, principal- amt, year, and rate of
interest. Using constructors, initialize the principal-amt and rate of interest. Find the
gross amount after a given period of investment.

6) Write a C++ program to sort N numbers using swap as friend function.

7) Write a C++ program to create a class called STACK using an array of integers.
Implement the following operations by overloading + & - and display the contents
of stack.

i) \[s1=s1+ \text{element} \] where s1 is an object of the class STACK and element is an
integer to be pushed on to top of the stack.

ii) \[s1=s1- \] where s1 is an object of the class STACK and – operator pops the
element.

8) Write a C++ program to create a class NAME and implement the following
operations. Display the result after every operation by overloading the \(<<\).

i) NAME firstname = “Herbert”

ii) NAME lastname = “Schield”

iii) NAME fullname = firstname +lastname
    (Use copy constructor)
9) Write a C++ program to create a class called MATRIX using a two-dimensional array of integers. Implement the following operations by overloading the operator \( = = \) which checks the compatibility of two matrices \( m1 \) and \( m2 \) to be added and subtracted. Perform the addition and subtraction by overloading the operators + and – respectively. Display the results.

```cpp
if (m1 == m2)
{
    m3 = m1 + m2;
    m4 = m1 - m2;
} else display Error message.
```

10) Write a C++ program to demonstrate the function overloading by overloading the user defined function called ADD with variable type and number of arguments. (Eg. ADD(int,int), ADD(int, float), ADD(init,int, float) etc.)

11) Create three classes, namely, STUDENT, EXAM and RESULT. The STUDENT class has data members, Rollno, Name and Branch and the class EXAM inherits the STUDENT class with new own data members, marks scored in six subjects. Derive the RESULT class from the EXAM class and it has its own data members, total_marks. Write an interactive program to model this multilevel inheritance relationship.

12) Create a base class RESERVATION. Create the derived classes, namely, ADULT, SENIOR_CITIZEN, CHILD by multiple inheritance. The RESERVATION class has data members, Name_of_passenger, Age, Date_of_journey, Source, Destination, Ticket_charge. Write an interactive program to display the ticket charges depending upon the category of passenger.

(Note : Charge for Children = \( \frac{1}{2} \) of adult ticket charge. Senior citizen = \( \frac{1}{4} \) of adult ticket charge.)

13) Write a C++ program to create a pure virtual function and show its access from the object of a derived class through the pointer of base class.

14) Write a C++ program to sort a given array of \( N \) numbers using template function.

(Note : For SEE, students will be asked to do similar programs from PART-B only.)