# CS 2021- Object-Oriented Programming Lab

| Ex. No. | Date | Program |
|---|---|---|
| 1 | 26/07/22 | 1. Classes, Data Hiding, Abstraction, Encapsulation |
| 2 | 02/08/22 | 1. Constructors and Destructors |
| 3 | 16/08/22 | 1. this pointer, Friend Function, Inline Function, Default Parameters |
| 4 | 23/08/22 | 1. Dynamic Memory Allocation |
| 5 | 30/08/22 | 1. Operator Overloading |
| 6 | 06/09/22 | 1. Function Overloading |
| 7 | 23/09/22 | 1. Inheritance |
| 8 | 27/09/22 | 1. Virtual Function |
| 9 | 11/10/22 | 1. Exception Handling |
| 10 | 18/10/22 | 1. Standard Template Library |
| 11 | 25/10/22 | 1. Tkinter - Frames, and Buttons |
| 12 | 01/11/22 | 1. Tkinter - Frames, and Buttons |

**Ex No. 1**                    <u>Classes, Data Hiding, Abstraction, and Encapsulation</u>
**Date: 26-07-2022**

1. Write a C++ program to implement a stack ADT as a header file using class. Create a class for a stack with the following operations, push, pop, display, peek, isFull, and isEmpty functions. Define all the functions inside the stack class. The data members (variables declared inside the class) should use "Private" access specifiers and member functions (functions declared and defined within the scope of a class) should use "Public" access specifiers. Assume the stack stores integers and can store a maximum of 5 integers. Import the created stack header file into an "application.cpp" file. The application.cpp has the following menus,

    1. Get Number
    2. Digit in Text
    3. Reverse
    4. Palindrome Check
    5. Exit

    Note: Stack stores the digits of the number entered in option 1. Option 1 is mandatory before using options 2 to 4 (Boundary condition). If option 1 is selected multiple times, a new number is obtained from the user and the stack is reset with the digits of the last entered number.
    Option 1 gets a number from the user. Extracts the digits and pushes the digits to a stack object.
    Option 2 writes a number in text form (Eg: Input: 12345, Output: one two three four five.)
    Option 3 reverses the given input number. (Eg: 123, Output: 321)
    Option 4 checks if the given number is palindrome or not
    Option 5 exits the program.

**Ex No. 2**           <u>Constructors and Destructors</u>

**Date: 02-08-2022**

1. Write a C++ program to implement a time class. It has private hours, minutes, and seconds data members. There are two options to create a clock namely,
   a. Create
   b. Create and Set Time

   Creation option calls a default constructor that sets all the private members to 0. The second option gets the hour, minutes, and seconds value from the user; creates, and calls a default constructor to initialize all private data members. A destructor is called to destroy the memory allocated to time class objects once they go out of scope. The program has the following Menu,

   a. Set Time
   b. Print Standard Time (AM or PM)
   c. Print Universal Time (24 hours)
   d. Print Indian Start Time (24 hours)
   e. Print Pacific Standard Time (AM or PM)
   f. Exit

   Maintain proper boundary conditions and follow coding best practices.

**Ex No. 3**
**Date: 16-08-2022**

<u>this pointer, Friend Function, Inline Function, Default Parameters</u>

1. Write a C++ program with Area and Volume class. The Area class has side, length, breadth, and radius as private data members. It has a square, rectangle, and circle public function that calculates and returns the area of the respective shape. If the parameters of a shape are not given, it calculates the unit area of the respective shape.

   The Volume class has height as a private data member. It has cube, cuboid, sphere, compare cube, compare cuboid, and compare sphere public member functions. The compare volume function prints which among the two volumes is greater than the other.

   Utilize the concepts of this pointer, friend function, inline function, and default parameters wherever necessary (minimum 1 of each concept)

   Write a menu-driven program to calculate,
      1. Area of a Square
      2. Area of a Rectangle
      3. Area of a Circle
      4. Volume of a Cube
      5. Volume of a Cuboid
      6. Volume of a Sphere
      7. Compare Cube
      8. Compare Cuboids
      9. Compare Spheres

**Ex No. 4**
**Date: 23-08-2022**

<u>Dynamic Memory Allocation (new and delete)</u>

1. Write a C++ program to implement a doubly linked list with insert, delete, display, and merge operations. The insert operation gets a value and inserts in the correct sorted position (ascending sort). The delete operation takes a position value in which a node can be deleted. The display option will get a list head and print the list. The merge option takes two sorted DLLs and returns a sorted merged DLL in O(m + n) computational time, where m and n are the length of the input DLLs. Print all the 3 lists at the end.

Utilize dynamic memory management wherever necessary.

The program has the following menu
   1. Insert
   2. Delete
   3. Display
   4. Merge
   5. Exit

**Example for merge option:**

**Input :** head1: 5<->7<->9
       head2: 4<->6<->8
**Output :** 4<->5<->6<->7<->8<->9
**Explanation:** The output list is in sorted order.

**Input :** head1: 1<->3<->5<->7
       head2: 2<->4
**Output :** 1<->2<->3<->4<->5<->7
**Explanation:** The output list is in sorted order.

**Ex No. 5**
**Date: 30-08-2022**

<u>Operator Overloading</u>

1. Consider a class "coordinate" that has two private data members xnum and ynum. The data members correspond to points in the positive quadrant of a 2D coordinate space. If the user enters a negative value for any point, convert the point into a positive coordinate point (Hint: Negation operator). Write a C++ program to get three points from the user and check if the length between the three points constitute a right angle triangle. Utilize operator overloading wherever possible (at least four instances). Overloading the negation operator is mandatory.

   The program has the following menu
   1. Set Point
   2. Display Point
   3. Display Length (Note: Calculates the distance between two points and displays the length)
   4. Check Right Triangle
   5. Exit

**Ex No. 6**
**Date: 06-09-2022**

<u>Function Overloading</u>

1. Write a C++ program to perform insert, delete, display, and search operations on a circular linked list. The insertion and deletion can happen in the beginning, end, or anywhere in between. Utilize function overloading wherever necessary (At least for 2 different operations). Maintain proper boundary conditions and code modularly. The menu for the program is,
   a. Insert
   b. Delete
   c. Search
   d. Display
   e. Exit

**Ex No. 7**
**Date: 23-09-2022**

<u>Inheritance</u>

1. Create an inheritance hierarchy that a bank might use to represent customers' bank accounts. All customers at this bank can deposit (i.e., credit) and withdraw money from their Accounts. An account has a unique ID number that is used to identify an account holder. There are two types of sub-accounts namely, Savings and Checking accounts. Savings accounts earn interest on the money they hold. Checking accounts, on the other hand, charges a constant fee per transaction (i.e., credit or debit). A search option searches for a customer ID and it prints all the saving and checking out details of that customer if present. A customer ID can at max have 1 saving and 1 checking account. Assume an account number is a 3-digit number. An account number for a savings account will be 1xx and for a checking account will be 2xx, where x can take any digit between 0 to 9. The account numbers are unique. Identify the data members, member function, base, and derived classes. The data about the customers are stored in an array of objects. Use public inheritance for derived classes. Maintain proper boundary conditions.

   Write a menu-driven program with the following options,

   1. Open Account (Savings or Checking Account)
   2. Credit (Savings or Checking Account)
   3. Debit (Savings or Checking Account)
   4. Change/Update Interest rate (Savings Account)
   5. Calculate Interest (Savings Account - Print)
   6. Calculate and Update the Principle Amount with Interest (Savings Account - Credit)
   7. Change/Update Fee Amount (Checking Account)
   8. Print Checking Fee (Checking Account)

9. Search Customer Details
10. Exit

[Question to self: What inheritance should be used and why?]

**Ex No. 8**
**Date: 27-09-2022**

<u>Virtual Functions</u>

Consider 4 classes **student, pg_student, research_student, and TA**. The **student** class has name and roll_number as data members. It has a **pure virtual** display member function. The **pg_student** class has course1, course2, and course3 data members. It also has a display member function that displays the name, roll_number, course1, 2, and 3 credited by the **pg_student**. The **research_student** class has course1 and research_area data members. It has a display function that displays the name, roll_number, course, and research_area of the **research_student**. The TA class has course_assigned and role data members. The 'role' can be either a "PG student" or a "Research student". It also has a display that displays the name, roll_number, course_assigned, and role. Based on the 'role', the display function should additionally display course1, 2, and 3 if the role was a "PG student", and course, and research_area if the role is a "Research student". Also, the course_assigned should not be the same as course1, course2, and course3 for a **pg_student**, and course_assigned should not be the same as the course for a **research_student**.

Identify the inheritance required and use the virtual keywords wherever necessary. For simplicity, create arrays of objects and set details only for the **TA** class alone. Maintain proper boundary conditions and follow coding best practices.

Write a menu-driven program with the following menus:

1. Set Details
2. Display
3. Exit

**Ex No. 9**
**Date: 11-10-2022**

<div align="center">

Exception Handling
</div>

1. Write a C++ program to convert an infix to a postfix expression with queues implemented using a singly linked list. Handle all boundary conditions using exception handling (try-catch-throw). Implement the following Menus

    1. Get Infix Expression
    2. Print Infix Expression
    3. Print Postfix Expression
    4. Exit

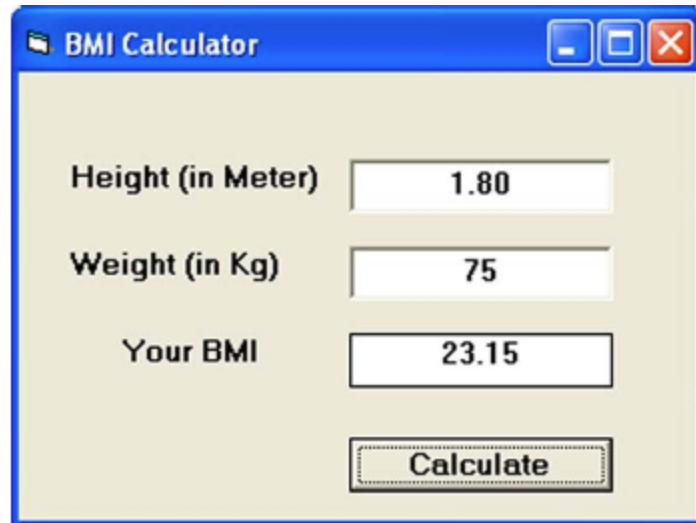    Hint: What data structure do you use to convert infix to postfix?
    **Note:**
1. Do not use STL
2. No brackets are allowed in infix expressions
3. Implement for +, -, *, and / operators only.
4. Precedence of +, - is less than the precedence of *, /
5. Precedence + and - has the same precedence, * and / have the same precedence
6. Associativity - left to right for all operators
7. Minimum 2 exception handling statements. Handling of wrong infix expressions is mandatory

**Ex No. 10**
**Date: 18-10-2022**

Standard Template Library (STL)

1. Write a C++ program to convert an infix to a postfix expression with queues implemented using a singly linked list. Handle all boundary conditions using exception handling (try-catch-throw). Implement the following Menus

    1. Get Infix Expression
    2. Print Infix Expression
    3. Print Postfix Expression
    4. Exit

    Hint: What data structure do you use to convert infix to postfix?
    **Note:**
1. **Use only STL**
2. No brackets are allowed in infix expressions
3. Implement for +, -, *, and / operators only.
4. Precedence of +, - is less than the precedence of *, /
5. Precedence + and - has the same precedence, * and / have the same precedence
6. Associativity - left to right for all operators
7. Minimum 1 exception handling statement. Handling of wrong infix expressions is mandatory

**Ex No. 11**

**Date: 29-10-2022**

<u>Tkinter - Frames and Buttons</u>

1. Write a Python 3 program to implement a simple BMI calculator using Tkinter.

**Ex No. 12**

**Date: 01-11-2022**

<u>Tkinter - Frames and Buttons</u>

1. Write a Python 3 program to implement a Marksheet and GPA calculator using Tkinter.

Grades

S - 10

A - 9

B - 8

C - 7

D - 6

E - 5

U, W, I - 0

Credit obtained  = Grade point x Sub Credit

Total Credit = Sum of all credits obtained per individual subject

SGPA =  Total Credits/ 15