Slip 1

Q1.Write the definition for a class Cylinder that contains data members radius and height. The class has the following member functions:

a. void setradius(float) to set the radius of data member.

b. void setheight(float) to set the height of data member.

c. float volume() to calculate and return the volume of the cylinder.

Write a C++ program to create cylinder object and display its volume.

ANS;

```
#include <iostream.h>
#include<conio.h>

class Cylinder {
private:
   float radius;
   float height;
public:
   void setradius(float r) {
         radius = r;
           }

   void setheight(float h) {
         height = h;
   }

   float volume() {
         return 3.14* radius * radius * height;
   }
};

void main() {
   Cylinder cyl;
   cyl.setradius(5.0);
   cyl.setheight(10.0);
   cout << "Volume of the cylinder: " << cyl.volume() << endl;
   getch();
}
```

Q2. Write a C++ program to create a class Array that contains one float array as member. Overload the Unary ++ and -- operators to increase or decrease the value of each element of an array. Use friend function for operator function

ANS:

```
#include <iostream.h>
#include<conio.h>

class Array {
private:
   float arr[5];

public:
   Array(float values[]) {
         for (int i = 0; i < 5; ++i) {
         arr[i] = values[i];
      }
   }
   friend Array operator++(Array &a) {
      for (int i = 0; i < 5; ++i) {
           ++a.arr[i];
         }
      return a;
   }
   friend Array operator--(Array &a) {
      for (int i = 0; i < 5; ++i) {
         --a.arr[i];
      }
      return a;
   }

   void display() {
      cout << "Array elements: ";
      for (int i = 0; i < 5; ++i) {
         cout << arr[i] << " ";
      }
      cout << endl;
   }
};

void main() {
   float values[] = {1.1, 2.2, 5.3, 4.4, 3.7};
     clrscr();
```

```
    Array arr(values);

    cout << "Original array:" << endl;
    arr.display();

    ++arr;
    cout << "After incrementing:" << endl;
    arr.display();

    --arr;
    cout << "After decrementing:" << endl;
    arr.display();
   getch();
}
```

Q2. Write a C++ program to create a class Shape with functions to find area of the shape and display the name of the shape and other essential components of the class. Create derived classes circle, rectangle and trapezoid each having overridden function area and display. Write a suitable program to illustrate Virtual Function.
ANS:

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include<string.h>
class Shape {
protected:
    char name[20];
public:
    Shape(char* n) {
        strcpy(name, n);
    }
    virtual void display() {
        cout << "Shape: " << name << endl;
    }
    virtual float area() {
        return 0.0;
    }
};

class Circle : public Shape {
private:
    float radius;
public:
    Circle(char* n, float r) : Shape(n), radius(r) {}
    void display() {
        Shape::display();
        cout << "Radius: " << radius << endl;
    }
    float area() {
        return 3.1415 * radius * radius;
    }
};

class Rectangle : public Shape {
private:
    float length, width;
public:
    Rectangle(char* n, float l, float w) : Shape(n), length(l), width(w) {}
    void display() {
        Shape::display();
        cout << "Length: " << length << ", Width: " << width << endl;
    }
    float area() {
        return length * width;
    }
};

class Trapezoid : public Shape {
private:
    float base1, base2, height;
public:
    Trapezoid(char* n, float b1, float b2, float h) : Shape(n), base1(b1), base2(b2), height(h) {}
    void display() {
        Shape::display();
        cout << "Base1: " << base1 << ", Base2: " << base2 << ", Height: " << height << endl;
    }
    float area() {
        return 0.5 * (base1 + base2) * height;
    }
};

int main() {
    clrscr();
    Shape* shapes[3];
    shapes[0] = new Circle("Circle", 5.0);
    shapes[1] = new Rectangle("Rectangle", 4.0, 6.0);
    shapes[2] = new Trapezoid("Trapezoid", 3.0, 5.0, 4.0);

    for (int i = 0; i < 3; i++) {
        shapes[i]->display();
        cout << "Area: " << shapes[i]->area() << endl;
        cout << endl;
    }

    for ( i = 0; i < 3; i++) {
        delete shapes[i];
    }

    getch();
    return 0;
}
```

Slip 2

Q1.Write a C++ program to create two classes Rectanglel and Rectangle. Compare area of both the rectangles using friend function.

ANS:

```
#include <iostream.h>
#include<conio.h>
class Rectangle2;
class Rectangle1 {
private:
    float length;
    float width;

public:
    Rectangle1(float l, float w) : length(l), width(w) {}

    float area() {
        return length * width;
    }

    friend void compareAreas(Rectangle1 &r1, Rectangle2 &r2);
};

class Rectangle2 {
```

```
private:
   float length;
   float width;

public:
   Rectangle2(float l, float w) : length(l), width(w) {}

   float area() {
         return length * width;
   }

   friend void compareAreas(Rectangle1 &r1, Rectangle2 &r2);
};

void compareAreas(Rectangle1 &r1, Rectangle2 &r2) {
   float area1 = r1.area();
   float area2 = r2.area();

   if (area1 > area2) {
         cout << "Area of Rectangle1 is greater than Rectangle2." << endl;
   } else if (area1 < area2) {
         cout << "Area of Rectangle2 is greater than Rectangle1." << endl;
   } else {
         cout << "Both rectangles have equal area." << endl;
   }
}

void main() {
   Rectangle1 rect1(5, 4);
   Rectangle2 rect2(6, 3);

   compareAreas(rect1, rect2);
  getch();
}
```

Q2. A book (ISBN) and CD (data capacity) are both types of media (I'd title) objects. A person buys 10 media items each of which can be either book or CD. Display the list of all books and CD's bought. Define the classes and appropriate member functions to accept and display data. Use pointers and concept of polymorphism (Virtual Function)
ANS:

```
#include <iostream.h>
#include <string.h>
#include<conio.h>
```

```
const int MAX_ITEMS = 4;
class Media {
protected:
   char title[50];
public:
   virtual void accept() = 0;
   virtual void display() = 0;
};

class Book : public Media {
   char ISBN[14]; // Assuming ISBN is of length 13 + '\0'
public:
   void accept() {
         cout << "Enter title of the book: ";
         cin.getline(title, 50);
         cout << "Enter ISBN: ";
         cin.getline(ISBN, 14);
   }

   void display() {
         cout << "Title: " << title << endl;
         cout << "ISBN: " << ISBN << endl;
   }
};

class CD : public Media {
   int capacity;
public:
   void accept() {
         cout << "Enter title of the CD: ";
         cin.getline(title, 50);
         cout << "Enter data capacity (in MB): ";
         cin >> capacity;
         cin.ignore(); // Clear input buffer
   }

   void display() {
         cout << "Title: " << title << endl;
         cout << "Data Capacity: " << capacity << " MB" << endl;
   }
};

int main() {
   Media* items[MAX_ITEMS];
```

```
    cout << "Enter details for 4 media items:\n";
    for (int i = 0; i < MAX_ITEMS; ++i) {
        char choice;
        cout << "Is this item a book (b) or a CD (c)? ";
        cin >> choice;
        cin.ignore(); // Clear input buffer

        if (choice == 'b') {
            items[i] = new Book();
        } else if (choice == 'c') {
            items[i] = new CD();
        } else {
            cout << "Invalid choice. Please enter 'b' for book or 'c' for CD." << endl;
            --i; // Decrement i to re-enter the loop for the same item
            continue;
        }
        items[i]->accept();
    }
    cout << "\nList of Books and CDs Bought:\n";
    for ( i = 0; i < MAX_ITEMS; ++i) {
        cout << "Item " << i + 1 << ":\n";
        items[i]->display();
        cout << endl;
        delete items[i]; // Free dynamically allocated memory
    }
    getch();
    return 0;
}
```

Q3.Write a C++ program to copy the contents of one file to another

Slip 3

Q1.Write a C++ program to overload function Volume and find Volume of Cube, Cylinder and
Sphere.
Ans

```
#include <iostream.h>
#include <conio.h>
```

```
// Inline function to calculate the area of a circle
inline float areaCircle(float radius) {
    return 3.14159 * radius * radius;
}

// Inline function to calculate the area of a square
inline float areaSquare(float side) {
    return side * side;
}

// Inline function to calculate the area of a rectangle
inline float areaRectangle(float length, float width) {
    return length * width;
}

int main() {
    clrscr(); // Clear the screen

    // Input data
    float radius = 5.0;
    float side = 4.0;
    float length = 6.0;
    float width = 3.0;

    // Calculate and display area of circle
    cout << "Area of Circle with radius " << radius << ": " << areaCircle(radius) << endl;

    // Calculate and display area of square
    cout << "Area of Square with side " << side << ": " << areaSquare(side) << endl;

    // Calculate and display area of rectangle
    cout << "Area of Rectangle with length " << length << " and width " << width << ": " <<
areaRectangle(length, width) << endl;

    getch(); // Wait for a key press before exiting
    return 0;
}
```

Q2. Write a C++ program with Student as abstract class and create derive classes Engineering,
Medicine and Science having data member rollno and name from base class Student. Create

objects of the derived classes and access them using array of pointer of base class Student.
Ans

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>

class Student {
protected:
    int rollno;
    char name[50];
public:
    virtual void setData(int r, char n[]) = 0;
    virtual void displayData() = 0;
};

class Engineering : public Student {
public:
    void setData(int r, char n[]) {
        rollno = r;
        strcpy(name, n);
    }
    void displayData() {
        cout << "Engineering Student\n";
        cout << "Roll No: " << rollno << endl;
        cout << "Name: " << name << endl;
    }
};

class Medicine : public Student {
public:
    void setData(int r, char n[]) {
        rollno = r;
        strcpy(name, n);
    }
    void displayData() {
        cout << "Medicine Student\n";
        cout << "Roll No: " << rollno << endl;
        cout << "Name: " << name << endl;
    }
};
```

```
class Science : public Student {
public:
    void setData(int r, char n[]) {
        rollno = r;
        strcpy(name, n);
    }
    void displayData() {
        cout << "Science Student\n";
        cout << "Roll No: " << rollno << endl;
        cout << "Name: " << name << endl;
    }
};

int main() {
    clrscr();
    Student *students[3];
    Engineering eng;
    Medicine med;
    Science sci;

    eng.setData(101, "John");
    med.setData(102, "Emily");
    sci.setData(103, "Michael");

    students[0] = &eng;
    students[1] = &med;
    students[2] = &sci;

    for(int i = 0; i < 3; i++) {
        students[i]->displayData();
    }

    getch();
    return 0;
}
```

Q3. Create a class String which contains a character pointer (Use new and delete operator)
Write a C++ program to overload following operators
a. ! To reverse the case of each alphabet from given string.
b. [ ] To print a character present at specified index

[3:18 p.m., 2024-03-21] +91 84466 09365:

```
 #include<iostream.h>
#include<conio.h>
#include<string.h>

class String {
private:
    char *str;
public:
    String() {
        str = NULL;
    }

    String(const char *s) {
        int len = strlen(s);
        str = new char[len + 1];
        strcpy(str, s);
    }

    ~String() {
        if(str != NULL)
            delete[] str;
    }

    String(const String &s) {
        int len = strlen(s.str);
        str = new char[len + 1];
        strcpy(str, s.str);
    }

    void operator!() {
        for (int i = 0; str[i] != '\0'; i++) {
            if (isalpha(str[i])) {
                if (isupper(str[i]))
                    str[i] = tolower(str[i]);
                else
                    str[i] = toupper(str[i]);
            }
        }
    }

    char& operator[](int index) {
        return str[index];
    }

    void display() {
        cout << str;
    }
};

int main() {
    clrscr();
    String s("Hello World!");
    !s; // Reversing the case of each alphabet
    s.display(); // Displaying the modified string

    cout << endl;
    cout << "Character at index 6: " << s[6] << endl;

    getch();
    return 0;
}
```

slip 4

Q1.Write a C++ program to print area of circle, square and rectangle using inline function.

Ans

```
#include<iostream.h>
#include<conio.h>

#define PI 3.14159
#define AREA_CIRCLE(radius) (PI * radius * radius)
#define AREA_SQUARE(side) (side * side)
#define AREA_RECTANGLE(length, width) (length * width)

int main() {
    float radius, side, length, width;

    // Input for circle
    cout << "Enter radius of the circle: ";
    cin >> radius;
    cout << "Area of the circle: " << AREA_CIRCLE(radius) << endl;
    // Input for square
    cout << "Enter side of the square: ";
    cin >> side;
    cout << "Area of the square: " << AREA_SQUARE(side) << endl;

    // Input for rectangle
    cout << "Enter length of the rectangle: ";
    cin >> length;
    cout << "Enter width of the rectangle: ";
    cin >> width;
    cout << "Area of the rectangle: " << AREA_RECTANGLE(length, width) << endl;

    getch();
    return 0;
}
```

: Q2. Write a C++ program to create a class which contains two dimensional integer array of size m*n
Write a member function to display transpose of entered matrix.(Use Dynamic Constructor for
allocating memory and Destructor to free memory of an object).
Ans

```
: #include <iostream.h>
#include <conio.h>

class Matrix {
private:
    int **data;
    int rows;
    int cols;

public:
    // Constructor to allocate memory for the matrix
    Matrix(int m, int n) {
        rows = m;
        cols = n;
        data = new int*[rows];
            for (int i = 0; i < rows; i++) {
            data[i] = new int[cols];
        }
    }

    // Destructor to free allocated memory
    ~Matrix() {
        for (int i = 0; i < rows; i++) {
            delete[] data[i];
        }
        delete[] data;
    }

    // Function to input values into the matrix
    void inputMatrix() {
        cout << "Enter the elements of the matrix:" << endl;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                cin >> data[i][j];
            }
        }
    }

    // Function to display the transpose of the matrix
    void displayTranspose() {
        cout << "Transpose of the matrix:" << endl;
        for (int j = 0; j < cols; j++) {
            for (int i = 0; i < rows; i++) {
                cout << data[i][j] << "\t";
            }
            cout << endl;
        }
    }
};

int main() {
    clrscr(); // Clear the screen

    int m, n;
    cout << "Enter the number of rows and columns of the matrix: ";
    cin >> m >> n;

    // Create a Matrix object with dynamic memory allocation
    Matrix mat(m, n);

    // Input values into the matrix
    mat.inputMatrix();
    // Display the transpose of the matrix
    mat.displayTranspose();

    getch(); // Wait for a key press before exiting
    return 0;
}
```

: Q3. Create a base class Flight containing protected data members as Flight_no, Flight_Name. Derive
a class Route(Source, Destination) from class Flight. Also derive a class Reservation (no_seats,
class, fare, travel_date) from Route. Write a C++ program to perform the following necessary
functions.
a. Enter details of n reservations.
b. Display reservation details of Business class.
ans

```
: #include <iostream.h>
#include <conio.h>

class Flight {
protected:
    int Flight_no;
    char Flight_Name[50];

public:
    // Function to input flight details
    void inputFlightDetails() {
        cout << "Enter Flight Number: ";
        cin >> Flight_no;
        cout << "Enter Flight Name: ";
        cin.ignore(); // Ignore newline character
        cin.getline(Flight_Name, 50);
    }
};

class Route : public Flight {
protected:
    char Source[50];
    char Destination[50];
```

```
public:
    // Function to input route details
    void inputRouteDetails() {
        cout << "Enter Source: ";
        cin.ignore(); // Ignore newline character
        cin.getline(Source, 50);
        cout << "Enter Destination: ";
        cin.getline(Destination, 50);
    }
};

class Reservation : public Route {
private:
    int no_seats;
    char class_type[20];
    float fare;
    char travel_date[20];

public:
    // Function to input reservation details
    void inputReservationDetails() {
        cout << "Enter number of seats: ";
        cin >> no_seats;
        cout << "Enter class (Business/Economy): ";
        cin >> class_type;
        cout << "Enter fare: ";
        cin >> fare;
        cout << "Enter travel date: ";
        cin >> travel_date;
    }

    // Function to display reservation details of Business class
    void displayBusinessClass() {
        if (strcmp(class_type, "Business") == 0) {
            cout << "Flight Number: " << Flight_no << endl;
            cout << "Flight Name: " << Flight_Name << endl;
            cout << "Source: " << Source << endl;
            cout << "Destination: " << Destination << endl;
            cout << "Number of Seats: " << no_seats << endl;
            cout << "Class: " << class_type << endl;
            cout << "Fare: " << fare << endl;
            cout << "Travel Date: " << travel_date << endl;
        }
    }
```

```
};

int main() {
    clrscr(); // Clear the screen

    int n;
    cout << "Enter the number of reservations: ";
    cin >> n;

    // Create an array of Reservation objects
    Reservation* reservations = new Reservation[n];

    // Input details of reservations
    for (int i = 0; i < n; i++) {
        cout << "\nEnter details of reservation " << i + 1 << ":" << endl;
        reservations[i].inputFlightDetails();
        reservations[i].inputRouteDetails();
        reservations[i].inputReservationDetails();
    }

    // Display reservation details of Business class
    cout << "\nReservation details of Business class:" << endl;
    for (int i = 0; i < n; i++) {
        reservations[i].displayBusinessClass();
    }

    delete[] reservations; // Free allocated memory

    getch(); // Wait for a key press before exiting
    return 0;
}
```

Slip 5

Q1.Write a C++ program to create a class Mobile which contains data members as Mobile_Id,
Mobile_Name, Mobile_Price. Create and Initialize all values of Mobile object by using parameterized constructor. Display the values of Mobile object.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class Mobile {
```

```
private:
    int Mobile_Id;
    char Mobile_Name[50];
    float Mobile_Price;

public:
    // Parameterized constructor
    Mobile(int id, char name[], float price) {
        Mobile_Id = id;
        strcpy(Mobile_Name, name);
        Mobile_Price = price;
    }

    // Function to display Mobile details
    void displayMobileDetails() {
        cout << "Mobile ID: " << Mobile_Id << endl;
        cout << "Mobile Name: " << Mobile_Name << endl;
        cout << "Mobile Price: " << Mobile_Price << endl;
    }
};

void main() {
    clrscr();
    // Creating a Mobile object and initializing it using parameterized constructor
    Mobile mobile1(101, "iPhone 12", 999.99);

    // Displaying the values of the Mobile object
    mobile1.displayMobileDetails();

    getch();
}
```

Q2.Create a base class Student (Roll_No, Name) which derives two classes Theory and Practical.
Theory class contains marks of five Subjects and Practical class contains marks of two practical
subjects. Class Result (Total_Marks, Class) inherits both Theory and Practical classes. (Use
concept of Virtual Base Class) Write a menu driven program to perform the following functions:
a. Build a master table.
b. Display master table.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class Student {
protected:
   int Roll_No;
   char Name[50];

public:
   void getData() {
      cout << "Enter Roll No: ";
      cin >> Roll_No;
      cout << "Enter Name: ";
      cin >> Name;
   }
};

class Theory : virtual public Student {
protected:
   float marks_theory[5];

public:
   void getTheoryMarks() {
      cout << "Enter Marks for 5 Theory Subjects: ";
      for (int i = 0; i < 5; i++) {
         cin >> marks_theory[i];
      }
   }
};

class Practical : virtual public Student {
protected:
   float marks_practical[2];

public:
   void getPracticalMarks() {
      cout << "Enter Marks for 2 Practical Subjects: ";
      for (int i = 0; i < 2; i++) {
         cin >> marks_practical[i];
      }
   }
};
```

```
class Result : public Theory, public Practical {
private:
    float Total_Marks;
    char Class[10];

public:
    void calculateTotalMarks() {
        Total_Marks = 0;
        for (int i = 0; i < 5; i++) {
            Total_Marks += marks_theory[i];
        }
        for (int i = 0; i < 2; i++) {
            Total_Marks += marks_practical[i];
        }
    }

    void calculateClass() {
        if (Total_Marks >= 400)
            strcpy(Class, "First Class");
        else if (Total_Marks >= 300)
            strcpy(Class, "Second Class");
        else if (Total_Marks >= 200)
            strcpy(Class, "Third Class");
        else
            strcpy(Class, "Fail");
    }

    void displayResult() {
        cout << "Roll No: " << Roll_No << endl;
        cout << "Name: " << Name << endl;
        cout << "Total Marks: " << Total_Marks << endl;
        cout << "Class: " << Class << endl;
    }
};

int main() {
    clrscr();
    Result r;

    // Build a master table
    r.getData();
    r.getTheoryMarks();
    r.getPracticalMarks();
    r.calculateTotalMarks();
    r.calculateClass();

    // Display master table
    cout << "\n-----Master Table-----\n";
    r.displayResult();

    getch();
    return 0;
}
```

OR

Q2.Create a class Book containing Book_name, author and Price as a data member and write
necessary member functions for the following (use function overloading).
a. To Accept and display the Book Information.
b. Display book details of a given author
c. Display book details of specific price

Ans

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class Book {
    char book_name[50];
    char author[50];
    float price;

public:
    void accept_details();
    void display_all();
    void display_by_author(char[]);
    void display_by_price(float);
};

void Book::accept_details() {
    cout << "Enter Book Name: ";
    cin.getline(book_name, 50);
    cout << "Enter Author Name: ";
    cin.getline(author, 50);
    cout << "Enter Price: ";
    cin >> price;
```

```
}

void Book::display_all() {
    cout << "Book Name: " << book_name << endl;
    cout << "Author: " << author << endl;
    cout << "Price: " << price << endl;
}

void Book::display_by_author(char given_author[]) {
    if (strcmp(author, given_author) == 0) {
        cout << "Book Name: " << book_name << endl;
        cout << "Author: " << author << endl;
        cout << "Price: " << price << endl;
    }
}

void Book::display_by_price(float given_price) {
    if (price == given_price) {
        cout << "Book Name: " << book_name << endl;
        cout << "Author: " << author << endl;
        cout << "Price: " << price << endl;
    }
}

int main() {
    clrscr();
    Book books[3]; // Assuming 3 books for simplicity

    // Accepting details for each book
    for (int i = 0; i < 3; i++) {
        cout << "Enter details for Book " << i + 1 << endl;
        books[i].accept_details();
    }

    // Display all books
    cout << "All Books:" << endl;
    for ( i = 0; i < 3; i++) {
        books[i].display_all();
    }

    // Display books by a specific author
    char given_author[50];
    cout << "Enter author name to search: ";
    cin.ignore(); // Clear the input buffer
    cin.getline(given_author, 50);
    cout << "Books by author " << given_author << ":" << endl;
    for (i = 0; i < 3; i++) {
            books[i].display_by_author(given_author);
    }

    // Display books of a specific price
    float given_price;
    cout << "Enter price to search: ";
    cin >> given_price;
    cout << "Books with price " << given_price << ":" << endl;
    for (i = 0; i < 3; i++) {
        books[i].display_by_price(given_price);
    }

    getch();
    return 0;
}
```

Slip 6

Q1.Write a C++ program to implement a class printdata to overload print function as follows:
void print(int) - outputs value followed by the value of the integer.
Eg. print(10) outputs - value 10
void print(char *) – outputs value followed by the string in double quotes.
Eg. print("hi") outputs value "hi " [10]
Ans:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

class printdata {
public:
    // Overloaded function to print integer value
    void print(int value) {
        cout << "value " << value << endl;
    }

    // Overloaded function to print string value
    void print(char* str) {
        cout << "value \"" << str << "\"" << endl;
    }
};

int main() {
    clrscr();
    printdata obj;

    // Call the overloaded functions
    obj.print(10);
    obj.print("hi");

    getch();
    return 0;
}
```

Q2.Write a C++ program to design a class complex to represent complex number. The complex class
uses an external function (as a friend function) to add two complex number. The function should
return an object of type complex representing the sum of two complex Numbers. [20]
Ans:

```
#include <iostream.h>
#include <conio.h>

class Complex {
private:
    float real;
    float imag;

public:
    // Constructor to initialize complex number
    Complex(float r = 0.0, float i = 0.0) {
        real = r;
        imag = i;
    }

    // Friend function to add two complex numbers
    friend Complex add(const Complex& c1, const Complex& c2);

    // Function to display complex number
    void display() {
        cout << real << " + " << imag << "i";
    }
};

// Friend function definition to add two complex numbers
Complex add(const Complex& c1, const Complex& c2) {
    Complex temp;
    temp.real = c1.real + c2.real;
    temp.imag = c1.imag + c2.imag;
    return temp;
}

int main() {
    clrscr();

    Complex c1(3.5, 2.5);
    Complex c2(2.5, 1.5);
    Complex sum = add(c1, c2);

    cout << "First complex number: ";
    c1.display();
    cout << endl;

    cout << "Second complex number: ";
    c2.display();
    cout << endl;

    cout << "Sum of complex numbers: ";
    sum.display();
    cout << endl;

    getch();
    return 0;
}
```

OR

Q2. Design two base classes Employee (Name, Designation) and Project (Project_Id, title). Derive
a class Emp_Proj(Duration) from Employee and Project. Write a menu driven program to
a. Build a master table. Display a master table
b. Display Project details in the ascending order of duration. [20]
Ans:

Slip 7

Q1.Write a C++ program using class which contains two data members as type integer. Create and
initialize the objects using default constructor, parameterized constructor with default value.
Write a member function to display maximum from given two numbers for all objects. [10]
Ans:

```
#include <iostream.h>
#include <conio.h>

class Number {
private:
   int num1, num2;
public:
   // Default Constructor
   Number() {
      num1 = 0;
      num2 = 0;
   }

   // Parameterized Constructor
   Number(int n1, int n2) {
      num1 = n1;
      num2 = n2;
   }

   // Function to display maximum from given two numbers
   void displayMax() {
      cout << "Maximum of " << num1 << " and " << num2 << " is: ";
      cout << (num1 > num2 ? num1 : num2) << endl;
   }
};

int main() {
   clrscr();

   // Creating objects using default constructor and parameterized constructor
   Number obj1; // Default constructor
```

```
    Number obj2(10, 20); // Parameterized constructor

    // Displaying maximum for obj1 and obj2
    obj1.displayMax();
    obj2.displayMax();

    getch();
    return 0;
}
```

Q2.Create a class College containing data members as College_Id, College_Name, Establishment_year, University_Name. Write a C++ program with following functions
a. Accept n College details
b. Display College details of specified University
c. Display College details according to Establishment year (Use Array of Objects and Function Overloading). [20]
Ans:

```
#include<iostream.h>
#include<conio.h>

class College {
    int College_Id;
    char College_Name[50];
    int Establishment_year;
    char University_Name[50];

public:
    // Function to accept college details
    void acceptDetails() {
        cout << "Enter College Id: ";
        cin >> College_Id;
        cout << "Enter College Name: ";
        cin >> College_Name;
        cout << "Enter Establishment year: ";
        cin >> Establishment_year;
        cout << "Enter University Name: ";
        cin >> University_Name;
    }

    // Function to display college details
    void displayDetails() {
        cout << "College Id: " << College_Id << endl;
        cout << "College Name: " << College_Name << endl;
        cout << "Establishment year: " << Establishment_year << endl;
        cout << "University Name: " << University_Name << endl;
    }

    // Function to display college details of specified university
    void displayDetails(const char* uniName) {
        if(strcmp(University_Name, uniName) == 0) {
            displayDetails();
        }
    }

    // Function to display college details according to establishment year
    void displayDetails(int year) {
        if(Establishment_year == year) {
            displayDetails();
        }
    }
};

void main() {
    int n;
    cout << "Enter the number of colleges: ";
    cin >> n;

    College colleges[100];

    // Accepting college details
    cout << "Enter details of " << n << " colleges:\n";
    for(int i = 0; i < n; i++) {
        cout << "Enter details for college " << i + 1 << ":\n";
        colleges[i].acceptDetails();
    }

    // Displaying college details of specified university
    char uniName[50];
    cout << "Enter the University Name to display its colleges: ";
    cin >> uniName;
    cout << "Colleges under " << uniName << ":\n";
    for(int i = 0; i < n; i++) {
        colleges[i].displayDetails(uniName);
    }

    // Displaying college details according to establishment year
    int year;
    cout << "Enter the Establishment Year to display colleges established in that year: ";
    cin >> year;
    cout << "Colleges established in " << year << ":\n";
    for(int i = 0; i < n; i++) {
        colleges[i].displayDetails(year);
    }

    getch(); // To hold the output screen
}
```

OR

Q2.Create a class Matrix and Write a C++ program to perform following functions:
a. To accept a Matrix
b. To display a Matrix
c. Overload unary minus '–' operator to calculate transpose of a Matrix
d. Overload binary multiplication '*' operator to calculate multiplication of two matrices
Ans:

Slip 8

Q1.Write a C++ program to subtract two integer numbers of two different classes using friend
function. [10]
Ans:

```
#include<iostream.h>
#include<conio.h>

// Forward declaration of class TwoNumbers
class TwoNumbers;

// Class declaration for class OneNumber
class OneNumber {
    int num1;

public:
    // Constructor to initialize num1
    OneNumber(int n) {
        num1 = n;
    }

    // Declare friend function
    friend int subtract(OneNumber, TwoNumbers);
};

// Class declaration for class TwoNumbers
class TwoNumbers {
    int num2;

public:
    // Constructor to initialize num2
    TwoNumbers(int n) {
        num2 = n;
    }

    // Declare friend function
    friend int subtract(OneNumber, TwoNumbers);
};

// Friend function definition to subtract numbers from both classes
int subtract(OneNumber obj1, TwoNumbers obj2) {
    return obj1.num1 - obj2.num2;
}

void main() {
    clrscr(); // Clear the screen

    // Create objects of both classes
    OneNumber obj1(10);
    TwoNumbers obj2(5);

    // Subtract numbers and display the result
    cout << "Subtraction result: " << subtract(obj1, obj2);

    getch(); // To hold the output screen
}
```

Q2.Create a class String which contains a character pointer (Use new and delete operator).
Write a C++ program to overload following operators:

a. ! To reverse the case of each alphabet from given string
b. == To check equality of two strings [20]
Ans:

OR

Q2.Write a C++ program to create a class Date which contains three data members as dd,mm,yyyy.

Create and initialize the object by using parameterized constructor and display date in dd-month-
yyyy format. (Input: 19-12-2014 Output: 19-Dec-2014) Perform validation for month. [20]

Ans:

Slip 9

Q1. Write a C++ program to create a class Item with data members Item_code, Item_name, Item_Price.
Write member functions to accept and display item information and also display number of objects
created for a class. (Use Static data member and Static member function) [10]

```
#include<iostream.h>
#include<conio.h>

class Item {
private:
    int Item_code;
    char Item_name[50];
    float Item_Price;
    static int count;

public:
    // Constructor to initialize data members
    Item(int code = 0, const char *name = "", float price = 0.0) {
        Item_code = code;
        strcpy(Item_name, name);
        Item_Price = price;
        count++; // Increment count whenever a new object is created
    }

    // Static member function to display number of objects created
    static void displayCount() {
        cout << "Number of objects created: " << count << endl;
    }

    // Function to accept item information
    void acceptItem() {
        cout << "Enter item code: ";
        cin >> Item_code;
        cout << "Enter item name: ";
        cin >> Item_name;
        cout << "Enter item price: ";
        cin >> Item_Price;
    }

    // Function to display item information
    void displayItem() {
        cout << "Item Code: " << Item_code << endl;
        cout << "Item Name: " << Item_name << endl;
        cout << "Item Price: " << Item_Price << endl;
    }
};

// Initializing static data member count
int Item::count = 0;

void main() {
    clrscr(); // Clear the screen

    // Creating objects of class Item
    Item item1, item2, item3;

    // Accepting item information
    cout << "Enter details for item 1:" << endl;
    item1.acceptItem();
    cout << endl;

    cout << "Enter details for item 2:" << endl;
    item2.acceptItem();
    cout << endl;

    cout << "Enter details for item 3:" << endl;
    item3.acceptItem();
    cout << endl;
    // Displaying item information
    cout << "Item details:" << endl;
    item1.displayItem();
    item2.displayItem();
    item3.displayItem();

    // Displaying number of objects created
    Item::displayCount();

    getch(); // To hold the output screen
}
```

Q2. Create a Base class Train containing protected data members as Train_no, Train_Name. Derive a
class Route(Route_id, Source, Destination) from Train class. Also derive a class Reservation
(Number_of_Seats, Train_Class, Fare, Travel_Date) from Route. Write a C++ program to perform following necessary functions:
a. Enter details of n reservations
b. Display details of all reservations
c. Display reservation details of a specified Train class [20]
Ans:

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

class Train {
protected:
    int Train_no;
    char Train_Name[50];
public:
    Train(int no, char* name) {
        Train_no = no;
        strcpy(Train_Name, name);
    }
};
```

```
class Route : public Train {
protected:
    int Route_id;
    char Source[50];
    char Destination[50];
public:
    Route(int no, char* name, int id, char* src, char* dest)
        : Train(no, name), Route_id(id) {
            strcpy(Source, src);
            strcpy(Destination, dest);
        }
};

class Reservation : public Route {
private:
    int Number_of_Seats;
    char Train_Class[20];
    float Fare;
    char Travel_Date[20];
public:
    Reservation(int no, char* name, int id, char* src, char* dest,
                int seats, char* tClass, float fare, char* date)
        : Route(no, name, id, src, dest), Number_of_Seats(seats),
          Fare(fare) {
              strcpy(Train_Class, tClass);
              strcpy(Travel_Date, date);
          }

    void display() {
        cout << "Train Number: " << Train_no << endl;
        cout << "Train Name: " << Train_Name << endl;
        cout << "Route ID: " << Route_id << endl;
        cout << "Source: " << Source << endl;
        cout << "Destination: " << Destination << endl;
        cout << "Number of Seats: " << Number_of_Seats << endl;
        cout << "Train Class: " << Train_Class << endl;
        cout << "Fare: " << Fare << endl;
        cout << "Travel Date: " << Travel_Date << endl << endl;
    }

    char* getTrainClass() {
        return Train_Class;
    }
};
```

```
void main() {
    clrscr();
    Reservation reservations[50];
    int n;
    cout << "Enter the number of reservations: ";
    cin >> n;

    for (int i = 0; i < n; ++i) {
        int trainNo, routeId, seats;
        char trainName[50], source[50], destination[50], tClass[20], date[20];
        float fare;

        cout << "\nEnter details for Reservation " << i + 1 << ":" << endl;
        cout << "Train Number: ";
        cin >> trainNo;
        cout << "Train Name: ";
        fflush(stdin);
        gets(trainName);
        cout << "Route ID: ";
        cin >> routeId;
        cout << "Source: ";
        fflush(stdin);
        gets(source);
        cout << "Destination: ";
        fflush(stdin);
        gets(destination);
        cout << "Number of Seats: ";
        cin >> seats;
        cout << "Train Class: ";
        fflush(stdin);
        gets(tClass);
        cout << "Fare: ";
        cin >> fare;
        cout << "Travel Date: ";
        fflush(stdin);
        gets(date);

        Reservation res(trainNo, trainName, routeId, source, destination,
                        seats, tClass, fare, date);
        reservations[i] = res;
    }

    // Displaying all reservations
    cout << "\nDetails of all reservations:" << endl;
    for (int i = 0; i < n; ++i) {
        reservations[i].display();
    }

    // Displaying reservation details of a specified Train class
    char specifiedClass[20];
    cout << "\nEnter the Train Class to display reservation details: ";
    fflush(stdin);
    gets(specifiedClass);
    cout << "\nReservation details for Train Class '" << specifiedClass << "':" << endl;
    for (int i = 0; i < n; ++i) {
        if (strcmp(reservations[i].getTrainClass(), specifiedClass) == 0) {
            reservations[i].display();
        }
    }

    getch();
}
```

OR

Q2. Create a class Time which contains data members as: Hours, Minutes and Seconds. Write a
C++ program to perform following necessary member functions:

a. To read time
b. To display time in format like: hh:mm:ss
c. To add two different times (Use Objects as argument) [20]
Ans:

Slip 10

Q1.Write a C++ program to create a class employee containing salary as a data member. Write
necessary member functions to overload the operator unary pre and post decrement "--" for decrementing salary. [10]

Ans:

```
#include<iostream.h>
```

```
#include<conio.h>

class Employee {
private:
    double salary;
public:
    // Constructor
    Employee(double s) : salary(s) {}

    // Overloading unary pre-decrement operator (--salary)
    void operator--() {
        --salary;
    }

    // Overloading unary post-decrement operator (salary--)
    void operator--(int) {
        salary--;
    }

    // Function to get salary
    double getSalary() const {
        return salary;
    }
};

int main() {
    clrscr();

    Employee emp(50000);

    cout << "Initial Salary: " << emp.getSalary() << endl;

    --emp; // pre-decrement
    cout << "Salary after pre-decrement: " << emp.getSalary() << endl;

    emp--; // post-decrement
    cout << "Salary after post-decrement: " << emp.getSalary() << endl;

    getch();
    return 0;
}
```

Q2. Design a base class Product(Product _Id, Product _Name, Price). Derive a class Discount
(Discount_In_Percentage) from Product. A customer buys n Products. Calculate total price,
total discount and display bill using appropriate manipulators. [20]
Ans:
OR

Q2. Create a class String which contains a character pointer (Use new and delete operator). Write a
C++ program to overload following operators:
a. < To compare length of two strings
b. == To check equality of two strings
c. + To concatenate two strings [20]

Ans:

SLIP 11

Q1.Write a C++ program to read two float numbers. Perform arithmetic binary operations +,-,*,/ on
these numbers using inline function. Display the resultant value.

Ans:
```
#include <iostream.h>
#include<conio.h>
float add(float a, float b) {
   return a + b;
}
float subtract(float a, float b) {
   return a - b;
}
float multiply(float a, float b) {
   return a * b;
}
float divide(float a, float b) {
   if (b != 0)
         return a / b;
   else {
```

```
            cout << "Error! Division by zero." << endl;
            return 0;
    }
}

int main() {
    float num1, num2;

    // Read two float numbers
    cout << "Enter the first float number: ";
    cin >> num1;
    cout << "Enter the second float number: ";
    cin >> num2;

    // Perform arithmetic operations
    cout << "Addition: " << add(num1, num2) << endl;
    cout << "Subtraction: " << subtract(num1, num2) << endl;
    cout << "Multiplication: " << multiply(num1, num2) << endl;
    cout << "Division: " << divide(num1, num2) << endl;
    getch();
    return 0;
}
```

Q2. Create a base class Conversion. Derive three different classes Weight (Gram, Kilogram),
Volume (Milliliter, Liter), Currency (Rupees, Paise) from Conversion class. Write a program
to perform read, convert and display operations. (Use Pure virtual function)
Ans

```
#include <iostream.h>
#include<conio.h>
class Conversion {
public:
    // Pure virtual function for conversion
    virtual void Convert() = 0;

    // Pure virtual function for display
    virtual void Display() = 0;
};

class Weight : public Conversion {
private:
    float grams;

public:
    // Function to read weight in grams
    void Read() {
        cout << "Enter weight in grams: ";
        cin >> grams;
    }

    // Function to convert grams to kilograms
    void Convert() {
        grams /= 1000; // 1 kilogram = 1000 grams
    }

    // Function to display weight in kilograms
    void Display() {
        cout << "Weight in kilograms: " << grams << " kg" << endl;
    }
};

class Volume : public Conversion {
private:
    float milliliters;

public:
    // Function to read volume in milliliters
    void Read() {
        cout << "Enter volume in milliliters: ";
        cin >> milliliters;
    }

    // Function to convert milliliters to liters
    void Convert() {
        milliliters /= 1000; // 1 liter = 1000 milliliters
    }

    // Function to display volume in liters
    void Display() {
        cout << "Volume in liters: " << milliliters << " L" << endl;
    }
};

class Currency : public Conversion {
private:
    float rupees;

public:
    // Function to read amount in rupees
    void Read() {
        cout << "Enter amount in rupees: ";
        cin >> rupees;
    }

    // Function to convert rupees to paise
    void Convert() {
        rupees *= 100; // 1 rupee = 100 paise
    }

    // Function to display amount in paise
    void Display() {
        cout << "Amount in paise: " << rupees << " paise" << endl;
    }
};

int main() {
    Weight weight;
    Volume volume;
    Currency currency;

    // Weight Conversion
    cout << "Weight Conversion:" << endl;
    weight.Read();
    weight.Convert();
    weight.Display();

    // Volume Conversion
    cout << "\nVolume Conversion:" << endl;
    volume.Read();
    volume.Convert();
    volume.Display();

    // Currency Conversion
    cout << "\nCurrency Conversion:" << endl;
    currency.Read();
    currency.Convert();
    currency.Display();
     getch();
    return 0;
```

```
}
```

Q2.Write a C++ program to create a class Person that contains data members as Person_Name, City,
Mob_No. Write a C++ program to perform following functions:
a. To accept and display Person information
b. To search the Person details of a given mobile number
c. To search the Person details of a given city.

(Use Function Overloading)
Ans

```
#include<iostream.h>
#include<string.h>
#include<conio.h>

class Person {
private:
    char Person_Name[50];
    char City[50];
    char Mob_No[15];

public:
    // Function to accept person information
    void acceptPersonInfo() {
        cout << "Enter Person Name: ";
        cin.ignore(); // To clear input buffer
        cin.getline(Person_Name, 50);
        cout << "Enter City: ";
        cin.getline(City, 50);
        cout << "Enter Mobile Number: ";
        cin.getline(Mob_No, 15);
    }

    // Function to display person information
    void displayPersonInfo() {
        cout << "Person Name: " << Person_Name << endl;
        cout << "City: " << City << endl;
        cout << "Mobile Number: " << Mob_No << endl;
    }

    // Function to search person details by mobile number
    void searchPersonInfo(const char* mobile) {
        if (strcmp(Mob_No, mobile) == 0) {
            displayPersonInfo();
        }
    }

    // Function to search person details by city
    void searchPersonInfoByCity(const char* city) {
        if (strcmp(City, city) == 0) {
            displayPersonInfo();
        }
    }
};

int main() {
    int n;
    clrscr(); // Clear screen for Turbo C++
    cout << "Enter the number of persons: ";
    cin >> n;

    Person *persons = new Person[n]; // Dynamic allocation of array of Person objects

    // Accept details for each person
    for (int i = 0; i < n; ++i) {
        cout << "\nEnter details for Person " << i+1 << ":" << endl;
        persons[i].acceptPersonInfo();
    }

    // Display details for each person
    cout << "\nDetails of all persons:" << endl;
    for ( i = 0; i < n; ++i) {
        cout << "\nDetails for Person " << i+1 << ":" << endl;
        persons[i].displayPersonInfo();
    }

    char searchMobile[15], searchCity[50];

    // Search person details by mobile number
    cout << "\nEnter mobile number to search: ";
    cin >> searchMobile;
    cout << "\nPerson details with mobile number " << searchMobile << ":" << endl;
    for ( i = 0; i < n; ++i) {
        persons[i].searchPersonInfo(searchMobile);
    }

    // Search person details by city
    cout << "\nEnter city to search: ";
    cin.ignore(); // To clear input buffer
    cin.getline(searchCity, 50);
    cout << "\nPerson details in city " << searchCity << ":" << endl;
    for ( i = 0; i < n; ++i) {
        persons[i].searchPersonInfoByCity(searchCity);
    }

    delete[] persons; // Freeing memory allocated for array of Person objects

    getch(); // Wait for a key press before exiting
    return 0;
}
```

SLIP 12

Q1.Write a C++ program to accept length and width of a rectangle. Calculate and display perimeter
as well as area of a rectangle by using inline function.
Ans

```
#include <iostream.h>
#include<conio.h>
inline float calculateArea(float length, float width) {
    return length * width;
}

inline float calculatePerimeter(float length, float width) {
    return 2 * (length + width);
}

int main() {
    float length, width;

    // Accepting length and width of the rectangle
    cout << "Enter length of the rectangle: ";
    cin >> length;
    cout << "Enter width of the rectangle: ";
    cin >> width;
    // Calculating and displaying area and perimeter using inline functions
    cout << "Area of the rectangle: " << calculateArea(length, width) << endl;
    cout << "Perimeter of the rectangle: " << calculatePerimeter(length, width) << endl;
     getch();
    return 0;
}
```

Q2.Write a C++ program to create a class which contains single dimensional integer array of
given size. Define member function to display median of a given array. (Use Dynamic Constructor to allocate and Destructor to free memory of an object).
Ans

```
#include <iostream.h>
#include <stdlib.h>
#include<conio.h>
class Array {
private:
    int *arr;
    int size;

public:
    // Dynamic constructor to allocate memory
    Array(int s) {
        size = s;
        arr = new int[size];
    }

    // Destructor to free memory
    ~Array() {
        delete[] arr;
    }

    // Function to input elements into the array
    void input() {
        cout << "Enter " << size << " integers:" << endl;
        for (int i = 0; i < size; i++) {
            cin >> arr[i];
        }
    }

    // Function to sort the array in ascending order
    void sort() {
        for (int i = 0; i < size - 1; i++) {
          for (int j = 0; j < size - i - 1; j++) {
              if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
              }
          }
        }
    }

    // Function to display the median of the array
    void displayMedian() {
        sort(); // Sort the array first

        if (size % 2 == 0) {
          // If the array size is even, median is the average of middle two elements
          cout << "Median of the array: " << (arr[size / 2 - 1] + arr[size / 2]) / 2.0 << endl;
        } else {
          // If the array size is odd, median is the middle element
          cout << "Median of the array: " << arr[size / 2] << endl;
        }
    }
};

int main() {
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;

    Array array(size); // Creating an object of class Array with given size

    array.input(); // Inputting elements into the array
    array.displayMedian(); // Displaying the median of the array
    getch();
    return 0;
}
```

Q2. Implement the following class hierarchy:
Employee: code, ename, desg
Manager (derived from Employee): year_of_experience, salary
Define appropriate functions to accept and display details.
Create n objects of the manager class and display the records.

Write main function that uses the above class and its member functions.

Ans\

```
#include<iostream.h>
#include<string.h>
#include<conio.h>

class Employee {
protected:
    int code;
    char ename[50];
    char desg[50];

public:
    // Function to accept details of employee
    void acceptEmployeeDetails() {
        cout << "Enter employee code: ";
        cin >> code;
        cout << "Enter employee name: ";
        cin.ignore(); // To clear the input buffer
        cin.getline(ename, 50);
        cout << "Enter employee designation: ";
        cin.getline(desg, 50);
    }

    // Function to display details of employee
    void displayEmployeeDetails() {
        cout << "Employee Code: " << code << endl;
        cout << "Employee Name: " << ename << endl;
        cout << "Employee Designation: " << desg << endl;
    }
};

class Manager : public Employee {
private:
    int year_of_experience;
    float salary;

public:
    // Function to accept details of manager
    void acceptManagerDetails() {
        acceptEmployeeDetails(); // Call base class function to accept employee details
        cout << "Enter years of experience: ";
        cin >> year_of_experience;
        cout << "Enter salary: ";
        cin >> salary;
    }

    // Function to display details of manager
    void displayManagerDetails() {
        displayEmployeeDetails(); // Call base class function to display employee details
        cout << "Years of Experience: " << year_of_experience << endl;
        cout << "Salary: " << salary << endl;
    }
};

int main() {
    int n;
    clrscr(); // Clear screen for Turbo C++
    cout << "Enter the number of managers: ";
    cin >> n;

    Manager *managers = new Manager[n]; // Dynamic allocation of array of Manager objects

    // Accept details for each manager
    for (int i = 0; i < n; ++i) {
        cout << "Details for Manager " << i+1 << ":" << endl;
        managers[i].acceptManagerDetails();
    }

    // Display details for each manager
    cout << "\nDetails of all managers:" << endl;
    for ( i = 0; i < n; ++i) {
        cout << "Details for Manager " << i+1 << ":" << endl;
        managers[i].displayManagerDetails();
        cout << endl;
    }

    delete[] managers; // Freeing memory allocated for array of Manager objects
    getch(); // Wait for a key press before exiting
    return 0;
}
```

SLIP 13

Q1.Write a C++ program to implement a class 'student' to overload following functions as follows:
a. int maximum(int, int) – returns the maximum score of two students
b. int maximum(int *a, int arraylength) – returns the maximum score from an array 'a'

Ans:

```
#include<iostream.h>
#include<conio.h>

class student {
public:
   int maximum(int score1, int score2) {
      return (score1 > score2) ? score1 : score2;
   }

   int maximum(int *a, int arraylength) {
      int max_score = a[0];
         for (int i = 1; i < arraylength; i++) {
         if (a[i] > max_score) {
            max_score = a[i];
         }
      }
      return max_score;
   }
};

int main() {
   student s;

   // Example usage of the first overloaded function
   int max_score1 = s.maximum(85, 92);
   cout << "Maximum score between 85 and 92: " << max_score1 << endl;

   // Example usage of the second overloaded function
   int scores[] = {78, 89, 95, 82, 90};
   int arraylength = sizeof(scores) / sizeof(scores[0]);
   int max_score2 = s.maximum(scores, arraylength);
   cout << "Maximum score from the array: " << max_score2 << endl;
    getch();
   return 0;
}
```

Q2.Write a C++ program to read the contents of a text file. Count and display number of characters,
words and lines from a file. Find the number of occurrences of a given word present in a file.

Ans

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#include<conio.h>

int countWords(char *);

void main() {
    clrscr();
    ifstream file;
    char filename[100], word[100];

    // Accepting file name
    cout << "Enter file name: ";
    cin.getline(filename, 100);

    // Opening file
    file.open(filename);

    if (!file) {
        cout << "Error in opening file!";
        getch();
        return;
    }

    char ch;
    int charCount = 0, wordCount = 0, lineCount = 0;

    while (file.get(ch)) {
        charCount++;

        // Counting words
        if (ch == ' ' || ch == '\n' || ch == '\t') {
            wordCount++;
        }

        // Counting lines
        if (ch == '\n') {
            lineCount++;
        }
    }

    // If last word is not followed by space
    if (charCount > 0) {
        wordCount++;
    }

    file.close();

    // Displaying counts
    cout << "Number of characters: " << charCount << endl;
    cout << "Number of words: " << wordCount << endl;
    cout << "Number of lines: " << lineCount << endl;

    // Accepting word to find occurrences
    cout << "Enter word to find occurrences: ";
    cin.getline(word, 100);

    int occurrences = countWords(word);
    cout << "Occurrences of '" << word << "': " << occurrences << endl;

    getch();
}

int countWords(char *word) {
    ifstream file;
    char filename[100];
    int count = 0;
    // Accepting file name
    cout << "Enter file name: ";
    cin.getline(filename, 100);
    // Opening file
    file.open(filename);
    if (!file) {
        cout << "Error in opening file!";
        return 0;
    }

    char str[100];

    // Counting occurrences of word
    while (file >> str) {
        if (strcmp(str, word) == 0)
            count++;
    }

    file.close();
    return count;
}
```

Q2.Write a C++ program to read the contents of a text file. Count and display number of characters,
words and lines from a file. Find the number of occurrences of a given word present in a file.

Ans

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#include<conio.h>

int countWords(char *);

void main() {
    clrscr();
    ifstream file;
    char filename[100], word[100];

    // Accepting file name
    cout << "Enter file name: ";
    cin.getline(filename, 100);

    // Opening file
    file.open(filename);

    if (!file) {
        cout << "Error in opening file!";
        getch();
        return;
    }

    char ch;
    int charCount = 0, wordCount = 0, lineCount = 0;

    while (file.get(ch)) {
        charCount++;

        // Counting words
        if (ch == ' ' || ch == '\n' || ch == '\t') {
            wordCount++;
        }

        // Counting lines
        if (ch == '\n') {
            lineCount++;
        }
    }

    // If last word is not followed by space
    if (charCount > 0) {
        wordCount++;
    }

    file.close();

    // Displaying counts
    cout << "Number of characters: " << charCount << endl;
    cout << "Number of words: " << wordCount << endl;
    cout << "Number of lines: " << lineCount << endl;

    // Accepting word to find occurrences
    cout << "Enter word to find occurrences: ";
    cin.getline(word, 100);

    int occurrences = countWords(word);
    cout << "Occurrences of '" << word << "': " << occurrences << endl;

    getch();
}

int countWords(char *word) {
    ifstream file;
    char filename[100];
    int count = 0;
    // Accepting file name
    cout << "Enter file name: ";
    cin.getline(filename, 100);
    // Opening file
    file.open(filename);
    if (!file) {
        cout << "Error in opening file!";
        return 0;
    }

    char str[100];

    // Counting occurrences of word
    while (file >> str) {
        if (strcmp(str, word) == 0)
            count++;
    }

    file.close();
    return count;
}
```

SLIP 14

Q1. Write a C++ program to interchange values of two integer numbers (use call by reference).[10]

Ans

```
#include<iostream.h>
#include<conio.h>

// Function to swap values using call by reference
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

void main() {
    clrscr(); // Clear screen for Turbo C++

    int num1, num2;

    // Input two numbers
    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;

    // Display numbers before swapping
    cout << "Before swapping:" << endl;
    cout << "First number: " << num1 << endl;
    cout << "Second number: " << num2 << endl;

    // Swap numbers
    swap(num1, num2);

    // Display numbers after swapping
    cout << "\nAfter swapping:" << endl;
    cout << "First number: " << num1 << endl;
    cout << "Second number: " << num2 << endl;

    getch(); // Wait for a key press before exiting
}
```

Q2. Write a C++ program to define a class Bus with the following specifications: Bus No, Bus
Name, No of Seats, Starting point, Destination .Write a menu driven program by using appropriate manipulators to
a. Accept details of n buses.
b. Display all bus details.
c. Display details of bus from specified starting point to destination

Ans

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

const int MAX_BUSES = 100; // Maximum number of buses

class Bus {
private:
    int busNo;
    char busName[50];
    int noOfSeats;
    char startPoint[50];
    char destination[50];

public:
    void setDetails(int no, char name[], int seats, char start[], char dest[]) {
        busNo = no;
        strcpy(busName, name);
        noOfSeats = seats;
        strcpy(startPoint, start);
        strcpy(destination, dest);
    }

    void displayDetails() {
        cout << "Bus No: " << busNo << endl;
        cout << "Bus Name: " << busName << endl;
        cout << "No of Seats: " << noOfSeats << endl;
        cout << "Starting Point: " << startPoint << endl;
        cout << "Destination: " << destination << endl;
    }

    char* getStartPoint() {
        return startPoint;
    }

    char* getDestination() {
        return destination;
    }
};

int main() {
    Bus buses[MAX_BUSES]; // Array to store buses
    int choice, n;
    char start[50], dest[50];
    int numBuses = 0; // Number of buses currently stored

    do {
        clrscr();
        cout << "\nMenu:\n";
        cout << "1. Accept details of n buses\n";
        cout << "2. Display all bus details\n";
        cout << "3. Display details of bus from specified starting point to destination\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter the number of buses: ";
            cin >> n;
            for (int i = 0; i < n; ++i) {
                Bus bus;
                int busNo, seats;
                char busName[50], startPoint[50], destination[50];
                cout << "Enter details for Bus " << i + 1 << ":\n";
                cout << "Bus No: ";
                cin >> busNo;
                cout << "Bus Name: ";
                cin.ignore();
                cin.getline(busName, 50);
                cout << "No of Seats: ";
                cin >> seats;
                cout << "Starting Point: ";
                cin.ignore();
                cin.getline(startPoint, 50);
                cout << "Destination: ";
                cin.getline(destination, 50);
                bus.setDetails(busNo, busName, seats, startPoint, destination);
                buses[numBuses++] = bus;
            }
            break;

        case 2:
            cout << "\nAll Bus Details:\n";
            for (int i = 0; i < numBuses; ++i) {
                buses[i].displayDetails();
                cout << endl;
            }
            break;

        case 3:
            cout << "Enter starting point: ";
            cin.ignore();
            cin.getline(start, 50);
            cout << "Enter destination: ";
            cin.getline(dest, 50);
            cout << "\nBus Details from " << start << " to " << dest << ":\n";
            for (int i = 0; i < numBuses; ++i) {
                if (strcmp(buses[i].getStartPoint(), start) == 0 && strcmp(buses[i].getDestination(),
dest) == 0) {
                    buses[i].displayDetails();
                    cout << endl;
                }
            }
            break;

        case 4:
            cout << "Exiting program...\n";
            break;

        default:
            cout << "Invalid choice. Please enter a number between 1 and 4.\n";
        }
        getch(); // Pause after displaying the output
    } while (choice != 4);

    return 0;
}
```

Q2. Create a class Fraction that contains two data members as numerator and denominator.
Write a C++ program to overload following operators
a. ++ Unary (pre and post both)
b. << and >> Overload as friend functions

Ans

```
#include<iostream.h>
#include<conio.h>

class Fraction {
private:
    int numerator;
    int denominator;

public:
    // Constructor
    Fraction(int num = 0, int denom = 1) : numerator(num), denominator(denom) {}

    // Overloading the pre-increment operator (++fraction)
    Fraction operator++() {
        numerator += denominator;
        return *this;
    }

    // Overloading the post-increment operator (fraction++)
    Fraction operator++(int) {
        Fraction temp = *this;
        numerator += denominator;
        return temp;
    }

    // Overloading the insertion operator (<<) as a friend function
    friend ostream& operator<<(ostream& out, const Fraction& frac) {
        out << frac.numerator << "/" << frac.denominator;
        return out;
    }

    // Overloading the extraction operator (>>) as a friend function
    friend istream& operator>>(istream& in, Fraction& frac) {
        cout << "Enter numerator: ";
        in >> frac.numerator;
        cout << "Enter denominator: ";
        in >> frac.denominator;
        return in;
    }
};

int main() {
    clrscr(); // Clear screen for Turbo C++

    Fraction f1, f2;

    // Input fractions
    cout << "Enter fraction 1:" << endl;
    cin >> f1;

    cout << "Enter fraction 2:" << endl;
    cin >> f2;

    // Display original fractions
    cout << "\nOriginal Fraction 1: " << f1 << endl;
    cout << "Original Fraction 2: " << f2 << endl;

    // Perform pre-increment operation
    cout << "\nAfter pre-increment operation:" << endl;
    cout << "Fraction 1: " << ++f1 << endl;
    cout << "Fraction 2: " << ++f2 << endl;

    // Perform post-increment operation
    cout << "\nAfter post-increment operation:" << endl;
    cout << "Fraction 1: " << f1++ << endl;
    cout << "Fraction 2: " << f2++ << endl;

    getch(); // Wait for a key press before exiting
    return 0;
}
```

SLIP 15

Q1.Write a C++ program to check minimum and maximum of two integer number (use inline function and conditional operator)

Ans

```
#include<iostream.h>
#include<conio.h>

// Inline function to find minimum of two integers
inline int findMin(int a, int b) {
    return (a < b) ? a : b;
}

// Inline function to find maximum of two integers
inline int findMax(int a, int b) {
    return (a > b) ? a : b;
}

void main() {
    clrscr(); // Clear screen for Turbo C++

    int num1, num2;

    // Input two numbers
    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;

    // Find and display minimum and maximum using inline functions and conditional operator
    cout << "Minimum of " << num1 << " and " << num2 << " is: " << findMin(num1, num2) <<
endl;
```

```
    cout << "Maximum of " << num1 << " and " << num2 << " is: " << findMax(num1, num2) <<
endl;

    getch(); // Wait for a key press before exiting
}
```

Q2.Create a base class Conversion. Derive three different classes Weight (Gram, Kilogram), Volume
(Milliliter, Liter), Currency (Rupees, Paise) from Conversion class. Write a program to perform
read, convert and display operations. (Use Pure virtual function)

Ans

```
#include<iostream.h>
#include<conio.h>

class Conversion {
public:
    // Pure virtual function for reading input
    virtual void readInput() = 0;

    // Pure virtual function for conversion
    virtual void convert() = 0;

    // Pure virtual function for displaying result
    virtual void displayResult() = 0;
};

class Weight : public Conversion {
protected:
    double value;

public:
    // Constructor
    Weight() {
        value = 0.0;
    }

    // Function to read weight input
    void readInput() {
        cout << "Enter weight value: ";
        cin >> value;
    }

    // Pure virtual function for conversion
    virtual void convert() = 0;

    // Pure virtual function for displaying result
    virtual void displayResult() = 0;
};

class Gram : public Weight {
public:
    // Function to convert Gram to Kilogram
    void convert() {
        value /= 1000.0; // 1 kilogram = 1000 grams
    }

    // Function to display result
    void displayResult() {
        cout << "Value in Kilograms: " << value << " kg" << endl;
    }
};

class Kilogram : public Weight {
public:
    // Function to convert Kilogram to Gram
    void convert() {
        value *= 1000.0; // 1 kilogram = 1000 grams
    }

    // Function to display result
    void displayResult() {
        cout << "Value in Grams: " << value << " g" << endl;
    }
};

// Similarly, implement classes Volume and Currency following the same structure as Weight

int main() {
    clrscr(); // Clear screen for Turbo C++

    Gram g;
    g.readInput();
    g.convert();
    g.displayResult();
    Kilogram kg;
    kg.readInput();
    kg.convert();
    kg.displayResult();

    // Similarly, create objects and perform operations for Volume and Currency classes

    getch(); // Wait for a key press before exiting

    return 0;
}
```

Q2. Write C++ program to create a class Employee containing data members Emp_no, Emp_Name,
Designation and Salary. Create and initialize the objects using default, parameterized and Copy
Constructor. Also write member function to calculate Income tax of the employee which is 20%
of salary.

Ans

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class Employee {
private:
    int Emp_no;
    char Emp_Name[50];
    char Designation[50];
    double Salary;

public:
    // Default constructor
    Employee() {
        Emp_no = 0;
        strcpy(Emp_Name, "Unknown");
        strcpy(Designation, "Unknown");
        Salary = 0.0;
    }

    // Parameterized constructor
    Employee(int empNo, const char* empName, const char* designation, double salary) {
        Emp_no = empNo;
        strcpy(Emp_Name, empName);
        strcpy(Designation, designation);
        Salary = salary;
    }

    // Copy constructor
    Employee(const Employee& emp) {
        Emp_no = emp.Emp_no;
        strcpy(Emp_Name, emp.Emp_Name);
        strcpy(Designation, emp.Designation);
        Salary = emp.Salary;
    }

    // Function to calculate income tax (20% of salary)
    double calculateIncomeTax() {
        return 0.2 * Salary;
    }

    // Function to display employee details
    void displayDetails() {
        cout << "Employee No: " << Emp_no << endl;
        cout << "Employee Name: " << Emp_Name << endl;
        cout << "Designation: " << Designation << endl;
        cout << "Salary: " << Salary << endl;
        cout << "Income Tax: " << calculateIncomeTax() << endl;
    }
};

int main() {
    clrscr(); // Clear screen for Turbo C++

    // Default constructor
    Employee emp1;
    cout << "Default Constructor - Employee Details:" << endl;
    emp1.displayDetails();
    cout << endl;

    // Parameterized constructor
    Employee emp2(101, "John Doe", "Manager", 50000.0);
    cout << "Parameterized Constructor - Employee Details:" << endl;
    emp2.displayDetails();
    cout << endl;
    // Copy constructor
    Employee emp3 = emp2;
    cout << "Copy Constructor - Employee Details:" << endl;
    emp3.displayDetails();
    cout << endl;

    getch(); // Wait for a key press before exiting

    return 0;
}
```

SLIP 16

Q1. Write a C++ program to create a class Number which contains two integer data members. Create
and initialize the object by using default constructor, parameterized constructor. Write a member function to display maximum from given two numbers for all objects. [10]
Ans

```
#include<iostream.h>
#include<conio.h>

class Number {
private:
    int num1;
    int num2;

public:
    // Default constructor
    Number() {
        num1 = 0;
        num2 = 0;
    }

    // Parameterized constructor
    Number(int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }

    // Function to display maximum of the two numbers
    void displayMax() {
        cout << "Maximum of " << num1 << " and " << num2 << " is: ";
        if (num1 > num2)
            cout << num1;
        else
            cout << num2;
        cout << endl;
    }
};

int main() {
    // Using default constructor
    Number obj1;
    obj1.displayMax();

    // Using parameterized constructor
    Number obj2(5, 10);
    obj2.displayMax();

    getch(); // Wait for user input before closing the console
    return 0;
}
```

Q2. Create two base classes Learn_Info(Roll_No, Stud_Name, Class, Percentage) and Earn_Info(No_of_hours_worked, Charges_per_hour). Derive a class Earn_Learn_info from
above two classes. Write necessary member functions to accept and display Student information. Calculate total money earned by the student. (Use constructor in derived class)
[20]

```
#include<iostream.h>
#include<conio.h>

class Number {
private:
    int num1;
    int num2;

public:
    // Default constructor
    Number() {
        num1 = 0;
        num2 = 0;
    }

    // Parameterized constructor
    Number(int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }

    // Function to display maximum of the two numbers
    void displayMax() {
        cout << "Maximum of " << num1 << " and " << num2 << " is: ";
        if (num1 > num2)
            cout << num1;
        else
            cout << num2;
        cout << endl;
    }
};

int main() {
    // Using default constructor
    Number obj1;
    obj1.displayMax();

    // Using parameterized constructor
    Number obj2(5, 10);
    obj2.displayMax();

    getch(); // Wait for user input before closing the console
    return 0;
}
```

Ans

OR

Q2. Create a class Time containing members as:
- hours
- minutes
- seconds
Write a C++ program for overloading operators >> and << to accept and display a Time also write a member function to display time in total seconds. [20]

Ans

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

// Base class Learn_Info
class Learn_Info {
protected:
    int Roll_No;
    char Stud_Name[50];
    char Class[20];
    float Percentage;
public:
    // Parameterized Constructor
    Learn_Info(int roll, char* name, char* cls, float percent) {
        Roll_No = roll;
        strcpy(Stud_Name, name);
        strcpy(Class, cls);
        Percentage = percent;
    }

    // Function to display student information
    void displayInfo() {
        cout << "Student Information:" << endl;
        cout << "Roll No: " << Roll_No << endl;
        cout << "Name: " << Stud_Name << endl;
        cout << "Class: " << Class << endl;
        cout << "Percentage: " << Percentage << "%" << endl;
    }
};

// Base class Earn_Info
class Earn_Info {
protected:
    int No_of_hours_worked;
    float Charges_per_hour;
public:
    // Parameterized Constructor
    Earn_Info(int hours, float charges) {
        No_of_hours_worked = hours;
        Charges_per_hour = charges;
    }
};

// Derived class Earn_Learn_Info
class Earn_Learn_Info : public Learn_Info, public Earn_Info {
public:
    // Constructor initializing base class constructors
    Earn_Learn_Info(int roll, char* name, char* cls, float percent, int hours, float charges)
        : Learn_Info(roll, name, cls, percent), Earn_Info(hours, charges) {}

    // Function to calculate total money earned
    float totalEarned() {
        return No_of_hours_worked * Charges_per_hour;
    }
};

void main() {
    clrscr();

    // Creating object of derived class
    Earn_Learn_Info student(1234, "John", "12th", 85.5, 40, 10.0);

    // Displaying student information
    student.displayInfo();

    // Calculating and displaying total money earned
    cout << "Total Money Earned: $" << student.totalEarned() << endl;

    getch();
}
```

SLIP 17

Q1.Write a C++ program to check if number is prime or not. [10]
Ans

```
#include<iostream.h>
#include<conio.h>

bool isPrime(int num) {
    if (num <= 1) return false; // 0 and 1 are not prime
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return false; // If the number is divisible by any number less than or equal
to its square root, it's not prime
    }
    return true; // If the number is not divisible by any number less than or equal to its square
root, it's prime
}

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    if (isPrime(num)) {
        cout << num << " is a prime number." << endl;
    } else {
        cout << num << " is not a prime number." << endl;
    }

    getch();
    return 0;
}
```

Q2. Create a class Fraction containing data members as Numerator and Denominator. Write a program to overload operators ++ , -- and * to increment, decrement a Fraction and
multiply two Fraction respectively. (Use constructor to initialize values of an object) [20]
Ans

```
#include<iostream.h>
#include<conio.h>

class Fraction {
private:
    int numerator;
    int denominator;

public:
    // Constructor to initialize values
    Fraction(int num = 0, int denom = 1) {
        numerator = num;
        denominator = denom;
    }
    // Overloading prefix increment operator ++
    Fraction operator++() {
        numerator = numerator + denominator;
        return *this;
    }

    // Overloading prefix decrement operator --
    Fraction operator--() {
        numerator = numerator - denominator;
        return *this;
    }

    // Overloading multiplication operator *
    Fraction operator*(const Fraction& other) {
        Fraction result;
        result.numerator = this->numerator * other.numerator;
        result.denominator = this->denominator * other.denominator;
        return result;
    }

    // Function to display fraction
    void display() {
        cout << numerator << "/" << denominator << endl;
    }
};

int main() {
    Fraction fraction1(3, 4);
    Fraction fraction2(2, 5);

    cout << "Initial Fraction 1: ";
    fraction1.display();
    cout << "Initial Fraction 2: ";
    fraction2.display();

    cout << "Incrementing Fraction 1: ";
    ++fraction1;
    fraction1.display();

    cout << "Decrementing Fraction 2: ";
    --fraction2;
    fraction2.display();
    cout << "Multiplying Fraction 1 and Fraction 2: ";
    Fraction result = fraction1 * fraction2;
    result.display();

    getch();
    return 0;
}
```

OR

Q2. Create a base class Media. Derive two different classes Book (Book_id, Book_name, Publication, Author, Book_price) and CD (CD_title, CD_price) from Media. Write a program
to accept and display information of both Book and CD. (Use pure virtual function) [20]
Ans

SLIP 18

Q1.Write a C++ program to calculate following series: (1)+(1+2)+(1+2+3)+(1+2+3+4) +... +(1+2+3+4+...+n)
Ans:

```
#include <iostream.h>
#include<conio.h>

void main() {
    int n;
    cout << "Enter the value of n: ";
    cin >> n;

    int sum = 0;
    for (int i = 1; i <= n; i++) {
        int innerSum = 0;
        for (int j = 1; j <= i; j++) {
            innerSum += j;
        }
        sum += innerSum;
    }
    cout << "Sum of the series is: " << sum << endl;
    getch();
}
```

Q2. Create a class called LIST with two pure virtual function store() and retrieve(). To store a value call store and to retrieve call retrieves function. Derive two classes stack and queue from it and override store and retrieve.
ANS:

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>

struct node
{
    int data;
    node *next;
};

node *head=NULL,*tail=NULL;
class List
{
    public:
    void view()
    {
            node *n = head;
            if(head==NULL)
            {
                cout<<"\n No elements found...";
            }
            else
            {
                cout<<" ";
                while(n!=NULL)
                {
                        if(n->next==NULL)
                        {
                            cout<<n->data;
                        }
                        else
                        {
                            cout<<n->data<<"->";
                        }
                        n = n->next;
            }
        }
    }
    virtual void store(int n)=0;
    virtual int retrive()=0;
};
class Stack :public List
{
    public:
    void store(int n)
    {
        node *n1 = new node();
        n1->data = n;
        n1->next=NULL;
        if((head==NULL)&&(tail==NULL))
        {
            head = n1;
            tail = n1;
        }
        else
        {
            tail->next = n1;
            tail = n1;
        }
    }
    int retrive()
    {
        if((tail==NULL)&&(head==NULL))
        {
            return -1;
        }
        else
        {
            int n = tail->data;
            node *n1 = head;
            while((n1->next!=tail)&&(head!=tail))
            {
                n1 = n1->next;
            }
            n1->next = NULL;
            free(tail);
            if(head!=tail)
            {
                tail = n1;
                }
                else
                {
                        tail=NULL;
                        head=NULL;
                }
                return n;
            }
        }
};

class Queue:public List
{
    public:
    void store(int n)
    {
            node *n1 = new node();
            n1->data = n;
            n1->next=NULL;
            if((head==NULL)&&(tail==NULL))
            {
                head = n1;
                tail = n1;
            }
            else
            {
                tail->next = n1;
                tail = n1;
            }
    }
    int retrive()
    {
            if((tail==NULL)&&(head==NULL))
            {
                return -1;
            }
            else
            {
                int n = head->data;
                if(head==tail)
                {
                        head = tail = NULL;
                }
                else
            {
                  head = head->next;
            }
            return n;
         }
   }
};

int main()
{
   Stack s1;
   int ch;
   while(1)
   {
         system("cls");
         cout<<"\n\n Program to implement stack and queue using pure virtual functions store
and retrieve";
         cout<<"\n\n Menu";
         cout<<"\n\n 1. Stack";
         cout<<"\n 2. Queue";
         cout<<"\n 3. Exit";
         cout<<"\n\n Enter your choice - ";
         cin>>ch;
         if(ch==1)
         {
            Stack s1;
            int ch1;
            while(1)
            {
                  system("cls");
                  cout<<"\n\n Stack Menu";

                  cout<<"\n 1. Push Element";
                  cout<<"\n 2. Pop Element";
                  cout<<"\n 3. View Stack";
                  cout<<"\n 4. Exit";
                  cout<<"\n\n Enter your choice - ";
                  cin>>ch1;
                  if(ch1==1)
                  {
                     int element;
                     cout<<"\n Enter the element you want to push - ";
                     cin>>element;
                     s1.store(element);
```

```
                cout<<"\n Element Pushed";
            }
            else if(ch1==2)
            {
                int element=0;
                element = s1.retrive();
                if(element==-1)
                {
                        cout<<"\n Stack is Empty";
                }
                else
                {
                        cout<<"\n Element Popped = "<<element;
                }
            }
            else if(ch1==3)
            {
                cout<<"\n Elements in stack from bottom to top:- ";
                s1.view();
            }
            else if(ch1==4)
            {
                break;
            }
            else
            {
                cout<<"\n\n Wrong choice";
            }
            getch();
        }
    }
    else if(ch==2)
    {
        Queue q1;
        int ch1;
        while(1)
        {
            system("cls");
            cout<<"\n\n Queue Menu";

            cout<<"\n 1. Push Element";
            cout<<"\n 2. Pop Element";
            cout<<"\n 3. View Queue";
            cout<<"\n 4. Exit";
            cout<<"\n\n Enter your choice - ";
            cin>>ch1;
            if(ch1==1)
            {
               int element;
               cout<<"\n Enter the element you want to push - ";
               cin>>element;
               q1.store(element);
               cout<<"\n Element Pushed";
            }
            else if(ch1==2)
            {
               int element=0;
               element = q1.retrive();
               if(element==-1)
               {
                     cout<<"\n Queue is Empty";
               }
               else
               {
                     cout<<"\n Element Popped = "<<element;
               }
            }
            else if(ch1==3)
            {
               cout<<"\n Elements in queue from front to rear:- ";
               q1.view();
            }
            else if(ch1==4)
            {
               break;
            }
            else
            {
               cout<<"\n\n Wrong choice";
            }
            getch();
      }
   }
   else if(ch==3)
   {
      exit(0);
   }
   else
```

```
          {
             cout<<"\n\n Wrong Choice";
          }
          getch();
    }
    return 0;
}
```

Q2.Write a C++ program to read student information such as rollno, name and percentage of n students. Write the student information using file handling. [20]
Ans:

```
#include <iostream.h>
#include<conio.h>
#include <fstream.h>
struct student {
  int rollno;
  char name[50];
  float percentage;
};
void main() {
  int n;
  cout << "Enter the number of students: ";
  cin >> n;
  student *students = new student[n]; // Dynamically allocate memory
  // Read student information
  for (int i = 0; i < n; i++) {
    cout << "Enter student " << i + 1 << " information:" << endl;
    cout << "Roll number: ";
    cin >> students[i].rollno;
    cout << "Name: ";
    cin.ignore(); // Ignore previous newline character
    cin.getline(students[i].name, 50);
    cout << "Percentage: ";
    cin >> students[i].percentage;
  }
  // Write student information to a file
  ofstream outfile("student_information.txt");
  for ( i = 0; i < n; i++) {
    outfile << students[i].rollno << " " << students[i].name << " " << students[i].percentage <<
endl;
  }
  outfile.close();
  cout << "Student information has been written to the file student_information.txt" << endl;
  delete[] students; // Free dynamically allocated memory
```

```
 getch();
}
```

SLIP 19

Q1. Write a C++ program to display factors of a number.
Ans:

```
#include <iostream.h>
#include<conio.h>

void displayFactors(int num) {
    cout << "Factors of " << num << " are: ";
    for (int i = 1; i <= num; ++i) {
        if (num % i == 0) {
            cout << i << " ";
        }
    }
    cout << endl;
}

void main() {
    int number;
    cout << "Enter a positive integer: ";
    cin >> number;

    if (number <= 0) {
        cout << "Invalid input. Please enter a positive integer." << endl;
    } else {
        displayFactors(number);
    }
getch();
}
```

Q2. Write a C++ program to create a text file which stores employee information as emp_id, emp_name, emp_sal). Write a menu driven program with the options

a. Append

b. Modify

c. Display

d. Exit

Ans:

```
#include <iostream.h>
#include <fstream.h>
#include<conio.h>

struct Employee {
    int emp_id;
    char emp_name[50];
    double emp_sal;
};

void appendEmployeeInfo();
void modifyEmployeeInfo();
void displayEmployeeInfo();

int main() {
    char choice;

    do {
        cout << "Menu:" << endl;
        cout << "a. Append" << endl;
        cout << "b. Modify" << endl;
        cout << "c. Display" << endl;
        cout << "d. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 'a':
                appendEmployeeInfo();
                break;
            case 'b':
                modifyEmployeeInfo();
                break;
            case 'c':
                displayEmployeeInfo();
                break;
            case 'd':
                cout << "Exiting program." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 'd');

    return 0;
}

void appendEmployeeInfo() {
    ofstream file;
    file.open("employee.txt", ios::app);
    Employee emp;

    cout << "Enter employee ID: ";
    cin >> emp.emp_id;
    cout << "Enter employee name: ";
    cin >> emp.emp_name;
    cout << "Enter employee salary: ";
    cin >> emp.emp_sal;

    file << emp.emp_id << " " << emp.emp_name << " " << emp.emp_sal << endl;

    file.close();
}

void modifyEmployeeInfo() {

}

void displayEmployeeInfo() {
    ifstream file;
    file.open("employee.txt");
    Employee emp;

    if (!file) {
        cout << "Unable to open file." << endl;
        return;
    }

    while (file >> emp.emp_id >> emp.emp_name >> emp.emp_sal) {
        cout << "Employee ID: " << emp.emp_id << ", Name: " << emp.emp_name << ", Salary: "
<< emp.emp_sal << endl;
    }

    file.close();
```

```
}
```

Q2. Design a two base classes Employee (Name, Designation) and Project(Project_Id, title). Derive a class Emp_Proj(Duration) from Employee and Project. Write a menu driven program to
a. Build a master table.
b. Display a master table
c. Display Project details in the ascending order of duration.
ANS:

```
#include <iostream.h>
#include <conio.h>
const int EMP_PROJ = 100;
const int NAME_LENGTH = 50;
const int PROJECT_ID_LENGTH = 20;
const int TITLE_LENGTH = 100;
class Employee {
public:
   char name[NAME_LENGTH];
   char designation[NAME_LENGTH];
};
class Project {
public:
   char project_id[PROJECT_ID_LENGTH];
   char title[TITLE_LENGTH];
};
class Emp_Proj : public Employee, public Project {
public:
   int duration;
};
Emp_Proj master_table[EMP_PROJ];
int num_projects = 0;
void build_master_table() {
   if (num_projects >= EMP_PROJ) {
      cout << "Maximum number of projects reached." << endl;
      return;
   }
   cout << "Enter employee name: ";
   cin >> master_table[num_projects].name;
   cout << "Enter employee designation: ";
   cin >> master_table[num_projects].designation;
   cout << "Enter project ID: ";
   cin >> master_table[num_projects].project_id;
   cout << "Enter project title: ";
```

```
    cin >> master_table[num_projects].title;
    cout << "Enter project duration: ";
    cin >> master_table[num_projects].duration;
    num_projects++;
}
void display_master_table() {
    cout << "\nMaster Table:" << endl;
    for (int i = 0; i < num_projects; i++) {
        cout << "Name: " << master_table[i].name << ", Designation: " <<
master_table[i].designation << endl;
        cout << "Project ID: " << master_table[i].project_id << ", Title: " << master_table[i].title <<
endl;
        cout << "Duration: " << master_table[i].duration << " days" << endl;
    }
}
void bubble_sort_by_duration() {
    for (int i = 0; i < num_projects - 1; i++) {
        for (int j = 0; j < num_projects - i - 1; j++) {
            if (master_table[j].duration > master_table[j + 1].duration) {

                Emp_Proj temp = master_table[j];
                master_table[j] = master_table[j + 1];
                master_table[j + 1] = temp;
            }
        }
    }
}
void display_projects_sorted_by_duration() {
    bubble_sort_by_duration();
    cout << "\nProjects sorted by duration:" << endl;
    display_master_table(); }
int main() {
    char choice;
    do {
        cout << "\nMenu:" << endl;
        cout << "a. Build master table" << endl;
        cout << "b. Display master table" << endl;
        cout << "c. Display projects sorted by duration" << endl;
        cout << "x. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 'a':
```

```
            build_master_table();
            break;
         case 'b':
            display_master_table();
            break;
         case 'c':
            display_projects_sorted_by_duration();
            break;
         case 'x':
            cout << "Exiting program." << endl;
            break;
         default:
            cout << "Invalid choice. Please try again." << endl;
      }
   } while (choice != 'x');
   getch();
   return 0;
}
```

SLIP 20

Q1.Write a C++ program to sort integer and float array elements in ascending order by using function overloading.

.Ans:

```
#include<iostream.h>
#include<conio.h>
class arr
{
  public:
  int n,i,j; // n=no of elements
  float a[10],temp; // a=array for no of elements
  void accept()
  {
         cout<<"\nEnter how many elements: ";
         cin>>n;
         for(i=0;i<n;i++)
         {
                 cout<<"\nEnter elements: ";
              cin>>a[i];
         }
         cout<<"\n============Array is==============\n\n";
         for(i=0;i<n;i++)
         {
                 cout<<a[i]<<"\t";
         }
         for(i=0;i<n;i++)
         {
            for(j=0;j<n-i-1;j++)
            {
            if(a[j]>a[j+1])
            {
                   temp=a[j];
                   a[j]=a[j+1];
                   a[j+1]=temp;
            }
            }
         }
         cout<<"\n===========Sorting Array is===========\n\n";
         for(i=0;i<n;i++)
         {
                 cout<<a[i]<<"\t";
         }
  }
};
int main()
{
 clrscr();
  arr r; //object created
  r.accept();
  getch();
  return(0);
}
```

Q2. Write a C++ program to create a class Department which contains data members as Dept_Id, Dept_Name, H.O.D., Number_Of_staff. Write necessary member functions to
 a. Accept details from user for 'n' departments and write it in a file "Dept.txt".
 b. Display details of department from a file.
ANS:+

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class dept
{
  int did,staff;
  static int c;
  char dname[10],hod[10];
  public:
  static void count()
  {
        c=c+1;
  }
  void accept()
  {
    cout<<"Enter Dept-Id: ";
    cin>>did;
    cout<<"Enter Dept-Name: ";
    cin>>dname;
    cout<<"Enter HOD: ";
    cin>>hod;
    cout<<"Enter No Of Staff: ";
    cin>>staff;
    ofstream out;
    out.open("File.txt",ios::app);
    out<<"\nDept-ID      : "<<did;
    out<<"\nDept-Name    : "<<dname;
    out<<"\nHOD          : "<<hod;
    out<<"\nNo Of Staff  : "<<staff;
    out.close();
  }
  void display()
  {
    char ch;
   ifstream fin;  // ifstream object is used to open a file for reading purpose only.
   fin.open("File.txt");
   while(!fin.eof()) //eof=end of file
   {
     ch=fin.get();
     cout<<ch;
   }
   fin.close();
  }
 static void dis()
  {
        cout<<"\nTotal object are : "<<c<<endl;
  }
};
int dept::c;
void main()
{
  int n,i,ch;
  do{
```

```
  cout<<"\n1.Accept Details\n2.Display\n3.Count no of objects\n4.Exit\nEnter your choice :- ";
  cin>>ch;
  switch(ch)
  {
        case 1:cout<<"Enter how many department you want to enter :- ";
  cin>>n;
  dept d[20];
  for(i=0;i<n;i++)
  {
        d[i].accept();
        d[i].count();
  }break;
  case 2:for(i=0;i<n;i++)
  {
        d[i].display();

  }break;
  case 3:d[n-1].dis(); break;
  }
  }while(ch!=4);
getch();
}
```

Q2. Write a C++ program to read the contents of a "Sample.txt" file. Store all the uppercase
characters in "Upper.txt", lowercase characters in "Lower.txt" and digits in "Digit.txt" files.
Change the case of each character from "Sample.txt" and store it in "Convert.txt" file.
ANS

```
#include <iostream.h>
#include <fstream.h>
#include <ctype.h>
int main() {
   ifstream inputFile("Sample.txt");
   ofstream upperFile("Upper.txt");
   ofstream lowerFile("Lower.txt");
   ofstream digitFile("Digit.txt");
   ofstream convertFile("Convert.txt");
   if (!inputFile || !upperFile || !lowerFile || !digitFile || !convertFile) {
      cerr << "Error: Unable to open input or output file(s)." << endl;
      return 1;
   }
   char ch;
```

```
    while (inputFile.get(ch)) {
        if (isupper(ch)) {
            upperFile << ch;
            convertFile << static_cast<char>(tolower(ch));
        } else if (islower(ch)) {
            lowerFile << ch;
            convertFile << static_cast<char>(toupper(ch));
        } else if (isdigit(ch)) {
            digitFile << ch;
            convertFile << ch;
        } else {
            convertFile << ch;
        }
    }
    inputFile.close();
    upperFile.close();
    lowerFile.close();
    digitFile.close();
    convertFile.close();
    cout << "Contents of Sample.txt categorized and converted successfully." << endl;
    cout << "Output files: Upper.txt, Lower.txt, Digit.txt, Convert.txt" << endl;
    // Keep the console window open
    cin.get();
    return 0;
}
```