Slip 2

Q.1) Write a C++ program to create an inline function that returns the length of a given string.

```
#include <iostream>
#include <string>
using namespace std;
// Inline function to return length of a string
inline int getStringLength(const string& str) {
return str.length();
}
int main() {
string input;
// Get string input from user
cout << "Enter a string: ";
getline(cin, input); // Accepts spaces too
// Call inline function and display result
cout << "Length of the string = " << getStringLength(input) << endl;
return 0;
}
```

Q.2) 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 and ending destination by user.

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

class Bus {
    int Bus_No;
    string Bus_Name;
    int No_of_Seats;
    string Starting_point;
    string Destination;

public:
    // Function to accept bus details
    void accept() {
        cout << "Enter Bus Number: ";
        cin >> Bus_No;
        cin.ignore(); // to clear newline from input buffer
        cout << "Enter Bus Name: ";
        getline(cin, Bus_Name);

        cout << "Enter Number of Seats: ";
        cin >> No_of_Seats;
        cin.ignore();

        cout << "Enter Starting Point: ";
        getline(cin, Starting_point);

        cout << "Enter Destination: ";
        getline(cin, Destination);
    }

    // Function to display bus details
    void display() const {
        cout << setw(10) << Bus_No
             << setw(15) << Bus_Name
             << setw(10) << No_of_Seats
             << setw(15) << Starting_point
             << setw(15) << Destination << endl;
    }

    // Function to check for matching route
    bool matchRoute(const string &start, const string &end) const {
        return (Starting_point == start && Destination == end);
    }
};

int main() {
```

```
int n, choice;
cout << "Enter number of buses: ";
cin >> n;

Bus *buses = new Bus[n];

do {
    cout << "\n---- MENU ----\n";
    cout << "1. Accept details of buses\n";
    cout << "2. Display all bus details\n";
    cout << "3. Display bus details by route\n";
    cout << "4. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
    case 1:
        for (int i = 0; i < n; i++) {
            cout << "\nEnter details of Bus " << i + 1 << ":\n";
            buses[i].accept();
        }
        break;

    case 2:
        cout << "\n" << setw(10) << "Bus No"
             << setw(15) << "Bus Name"
             << setw(10) << "Seats"
             << setw(15) << "Start"
             << setw(15) << "Destination" << endl;
        cout << string(65, '-') << endl;
    for (int i = 0; i < n; i++) {
        buses[i].display();
    }
    break;

case 3: {
    cin.ignore();
    string start, end;
    cout << "Enter Starting Point: ";
    getline(cin, start);
    cout << "Enter Destination: ";
    getline(cin, end);

    cout << "\n" << setw(10) << "Bus No"
         << setw(15) << "Bus Name"
         << setw(10) << "Seats"
         << setw(15) << "Start"
         << setw(15) << "Destination" << endl;
    cout << string(65, '-') << endl;

    bool found = false;
    for (int i = 0; i < n; i++) {
        if (buses[i].matchRoute(start, end)) {
            buses[i].display();
            found = true;
        }
    }
    if (!found) {
        cout << "No bus found for the given route.\n";
    }
```

```
            break;
        }

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

        default:
            cout << "Invalid choice! Try again.\n";
        }

    } while (choice != 4);

    delete[] buses;
    return 0;
}
```

Slip 3

Q.1) Write a C++ program that reads Book.txt file and displays Books data on the screen.

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream file("Book.txt");   // open file

    if (!file) {
        cout << "File not found!" << endl;
        return 0;
    }

    char ch;
    while (file.get(ch)) {       // read character by character
        cout << ch;
    }

    file.close();
    return 0;
}
```

Q.2) Write the definition for a class called 'Point' that has x & y as integer data members. Use copy constructor to copy one object to another. (Use Default and parameterized constructor to initializethe appropriate objects)

```
#include <iostream>
using namespace std;
class Point {
int x, y;
public:
// Default constructor
Point() {
x = 0;
y = 0;
}
// Parameterized constructor
Point(int a, int b) {
x = a;
y = b;
}
// Copy constructor
Point(const Point &p) {
x = p.x;
y = p.y;
}
// Function to display point
void display() {
cout << "(" << x << ", " << y << ")" << endl;
}
};
int main() {
// Object using default constructor
```

```
Point p1;
cout << "Point p1 (Default Constructor): ";
p1.display();
// Object using parameterized constructor
Point p2(10, 20);
cout << "Point p2 (Parameterized Constructor): ";
p2.display();
// Object using copy constructor
Point p3(p2);
cout << "Point p3 (Copy Constructor, copy of p2): ";
p3.display();
return 0;
```

Slip 6

Q.1) Write a C++ program using class to calculate simple interest amount. (Use parameterized constructor with default value for rate).

```
#include <iostream>
using namespace std;
class SimpleInterest {
float principal, time, rate, interest;
public:
// Parameterized constructor with default rate value (10%)
SimpleInterest(float p, float t, float r = 10.0) {
principal = p;
time = t;
rate = r;
// Formula: SI = (P * T * R) / 100
interest = (principal * time * rate) / 100;
}
// Function to display result
void display() {
cout << "Principal: " << principal << endl;
cout << "Time (years): " << time << endl;
cout << "Rate (%): " << rate << endl;
cout << "Simple Interest: " << interest << endl;
}
};
// Main function
int main() {
// Object with default rate (10%)
cout << "Case 1: Using default rate (10%)" << endl;
SimpleInterest si1(5000, 2);
si1.display();
cout << "\nCase 2: Using custom rate (12.5%)" << endl;
SimpleInterest si2(8000, 3, 12.5);
si2.display();
return 0;
}
```

Q.2) Create a class Date with members as dd, mm, yyyy. Write a C++ program for overloading operators >> and << to accept and display a Date.

```
#include <iostream>
using namespace std;

class Date {
  int dd, mm, yyyy;

public:
  // Overload >> for input
  friend istream& operator>>(istream &in, Date &d) {
    cout << "Enter day (dd): ";
    in >> d.dd;
    cout << "Enter month (mm): ";
    in >> d.mm;
    cout << "Enter year (yyyy): ";
    in >> d.yyyy;
    return in;
  }

  // Overload << for output
  friend ostream& operator<<(ostream &out, const Date &d) {
    out << d.dd << "/" << d.mm << "/" << d.yyyy;
    return out;
  }
};

int main() {
  Date d1;

  cout << "Enter a date:" << endl;
  cin >> d1;

  cout << "You entered: " << d1 << endl;

  return 0;
}
```

Slip7

Q.1) 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.

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
// Base class Product
class Product {
protected:
int productId;
string productName;
float price;
public:
Product(int id = 0, string name = "", float p = 0.0) {
productId = id;
productName = name;
price = p;
}
virtual void display() {
cout << setw(10) << productId
<< setw(15) << productName
<< setw(10) << fixed << setprecision(2) << price;
}
float getPrice() {
return price;
}
};
// Derived class Discount
class Discount : public Product {
```

```
float discountPercentage;
public:
Discount(int id = 0, string name = "", float p = 0.0, float d = 0.0)
: Product(id, name, p) {
discountPercentage = d;
}
float getDiscountAmount() {
return (price * discountPercentage) / 100.0;
}
float getFinalPrice() {
return price - getDiscountAmount();
}
void display() override {
Product::display();
cout << setw(10) << discountPercentage
<< setw(12) << fixed << setprecision(2) << getDiscountAmount()
<< setw(12) << fixed << setprecision(2) << getFinalPrice() << endl;
}
};
int main() {
int n;
cout << "Enter number of products: ";
cin >> n;
Discount *products = new Discount[n]; // dynamic array
int id;
string name;
float price, discount;
for (int i = 0; i < n; i++) {
cout << "\nEnter details for product " << i + 1 << ":\n";
cout << "Product Id: ";
cin >> id;
cout << "Product Name: ";
cin >> name;
cout << "Price: ";
cin >> price;
cout << "Discount %: ";
cin >> discount;
products[i] = Discount(id, name, price, discount);
}
float totalPrice = 0, totalDiscount = 0, finalAmount = 0;
cout << "\n================== BILL ==================\n";
cout << setw(10) << "ID"
<< setw(15) << "Name"
<< setw(10) << "Price"
<< setw(10) << "Disc(%)"
<< setw(12) << "Disc Amt"
<< setw(12) << "Final Amt" << endl;
cout << "------------------------------------------------------------\n";
for (int i = 0; i < n; i++) {
products[i].display();
totalPrice += products[i].getPrice();
totalDiscount += products[i].getDiscountAmount();
finalAmount += products[i].getFinalPrice();
}
cout << "------------------------------------------------------------\n";
cout << setw(35) << "TOTAL:"
<< setw(12) << fixed << setprecision(2) << totalDiscount
<< setw(12) << fixed << setprecision(2) << finalAmount << endl;
delete[] products; // free memory
return 0; }
```

Q.2) Write a C++ program that appends the contents of one file to another file.

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    string file1, file2;
    cout << "Enter source file name: ";
    cin >> file1;
    cout << "Enter destination file name: ";
    cin >> file2;

    ifstream src(file1, ios::in);         // open source file for reading
    ofstream dest(file2, ios::app);       // open destination file for appending

    if (!src) {
        cout << "Error: Cannot open source file!" << endl;
        return 1;
    }
    if (!dest) {
        cout << "Error: Cannot open destination file!" << endl;
        return 1;
    }

    string line;
    while (getline(src, line)) {
        dest << line << endl;  // append line to destination file
    }
    cout << "File appended successfully!" << endl;

    src.close();
    dest.close();

    return 0;
}
```

Slip 8

Q.1) Write a program to define a class ‘Rectangle’ having data members length and breadth. Accept this data for one object and display area and perimeter of rectangle.

```
#include <iostream>
using namespace std;
class Rectangle {
private:
float length, breadth;
public:
// Function to accept input
void input() {
cout << "Enter length: ";
cin >> length;
cout << "Enter breadth: ";
cin >> breadth;
}
// Function to calculate and display area
void displayArea() {
float area = length * breadth;
cout << "Area: " << area << endl;
}
// Function to calculate and display perimeter
void displayPerimeter() {
float perimeter = 2 * (length + breadth);
cout << "Perimeter: " << perimeter << endl;
}
};
int main() {
Rectangle rect; // Create an object of Rectangle
rect.input(); // Accept length and breadth
rect.displayArea(); // Display area
rect.displayPerimeter(); // Display perimeter
return 0;
}
```

Q.2) Write a program for combining two strings also show the execution of dynamic constructor. For this declare a class 'Mystring' with data members as name and length.

```
#include <iostream>
#include <cstring> // for strcpy, strcat
using namespace std;
class String {
char *name; // pointer for dynamic memory
int length;
public:
// Default constructor
String() {
length = 0;
name = new char[1]; // allocate 1 byte for '\0'
name[0] = '\0';
}
// Parameterized constructor (Dynamic Constructor)
String(const char *s) {
length = strlen(s);
name = new char[length + 1]; // allocate memory dynamically
strcpy(name, s);
}
// Copy constructor
String(const String &s) {
length = s.length;
name = new char[length + 1];
strcpy(name, s.name);
}
// Function to concatenate two strings
String combine(const String &s) {
String temp;
temp.length = length + s.length;
delete[] temp.name; // free default allocation
temp.name = new char[temp.length + 1];
strcpy(temp.name, name);
strcat(temp.name, s.name);
return temp;
}
// Function to display string
void display() {
cout << name << endl;
}
// Destructor
~String() {
delete[] name; // free allocated memory
}
};
int main() {
String s1("Hello");
String s2("World");
cout << "String 1: ";
s1.display();
cout << "String 2: ";
s2.display();
// Combine two strings
String s3 = s1.combine(s2);
```

```
cout << "Combined String: ";
s3.display();
return 0;
}
```

Slip 9

Q.1) Write a program to declare a class Product containing data members product_code , name and price. Accept and display this information for 2 objects.

```
#include <iostream>
using namespace std;
class Product {
int product_code;
char name[50];
float price;
public:
// Function to accept data
void accept() {
cout << "Enter product code: ";
cin >> product_code;
cout << "Enter product name: ";
cin >> name;
cout << "Enter product price: ";
cin >> price;
}
// Function to display data
void display() {
cout << "\nProduct Code: " << product_code << endl;
cout << "Product Name: " << name << endl;
cout << "Product Price: " << price << endl;
}
};
int main() {
Product p1, p2;
cout << "Enter details of Product 1:\n";
p1.accept();
cout << "\nEnter details of Product 2:\n";
p2.accept();
cout << "\n--- Displaying Product Details ---";
cout << "\nProduct 1:";
p1.display();
cout << "\nProduct 2:";
p2.display();
return 0;
}
```

Q.2) Write a C++ program to merge two files into a single file using file handling. Assuming that a text file named FIRST.TXT contains some text written into it, write a function named vowelwords(), that reads the file FIRST.TXT and creates a new file named SECOND.TXT, tocontain only those words from the file FIRST.TXT which start with a lower-case vowel (i.e., with'a','e','i','o','u'). For example, if the file FIRST.TXT contains Carry umbrella and overcoat when itrains. Then the file SECOND.TXT shall contain umbrella, and, overcoat, it.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

// Function to copy vowel starting words
void vowelwords() {
   ifstream fin("FIRST.TXT");
   ofstream fout("SECOND.TXT");

   if (!fin) {
      cout << "Error: Cannot open FIRST.TXT" << endl;
      return;
```

```
    }
    if (!fout) {
        cout << "Error: Cannot create SECOND.TXT" << endl;
        return;
    }

    string word;
    while (fin >> word) {
        char ch = word[0];  // first character of word
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            fout << word << " ";
        }
    }

    cout << "Vowel words copied to SECOND.TXT successfully!" << endl;

    fin.close();
    fout.close();
}

int main() {
    string file1, file2, mergedFile;

    cout << "Enter first file name: ";
    cin >> file1;
    cout << "Enter second file name: ";
    cin >> file2;
    cout << "Enter merged file name: ";
    cin >> mergedFile;
    ifstream f1(file1);
    ifstream f2(file2);
    ofstream fout(mergedFile);

    if (!f1 || !f2 || !fout) {
        cout << "Error: Cannot open files!" << endl;
        return 1;
    }

    string line;
    while (getline(f1, line))
        fout << line << endl;

    while (getline(f2, line))
        fout << line << endl;

    cout << "Files merged into " << mergedFile << " successfully!" << endl;

    f1.close();
    f2.close();
    fout.close();

    // Now perform vowel word extraction
    vowelwords();

    return 0;
}
```

Slip 10

Q.1) 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 usingparameterized constructor. Display the values of Mobile object where Mobile_price should be rightjustified with a precision of two digits.

```
#include <iostream>
#include <iomanip> // for setw, setprecision, fixed
using namespace std;
class Mobile {
int Mobile_Id;
string Mobile_Name;
float Mobile_Price;
public:
// Parameterized constructor
Mobile(int id, string name, float price) {
Mobile_Id = id;
Mobile_Name = name;
Mobile_Price = price;
}
// Function to display Mobile details
void display() {
cout << "Mobile ID : " << Mobile_Id << endl;
cout << "Mobile Name : " << Mobile_Name << endl;
// Right justified price with precision 2
cout << "Mobile Price: "
<< right << setw(10) << fixed << setprecision(2) << Mobile_Price <<
endl;
}
};
int main() {
```

```
// Creating object using parameterized constructor
Mobile m1(101, "Samsung Galaxy S21", 54999.50);
Mobile m2(102, "iPhone 14", 79999.99);
cout << "Details of Mobile 1:" << endl;
m1.display();
cout << "\nDetails of Mobile 2:" << endl;
m2.display();
return 0;
}
```

Q.2) Create a base class Shape. Derive three different classes Circle, Rectangle and Triangle from Shape class. Write a C++ program to calculate area of Circle, Rectangle and Triangle. (Use purevirtual function).

```
#include <iostream>
#include <cmath>   // for M_PI
using namespace std;

// Base class
class Shape {
public:
   virtual void area() = 0;   // Pure virtual function
};

// Derived class Circle
class Circle : public Shape {
   float radius;
public:
   Circle(float r) { radius = r; }
   void area() {
      cout << "Area of Circle = " << M_PI * radius * radius << endl;
   }
```

```
};

// Derived class Rectangle
class Rectangle : public Shape {
   float length, breadth;
public:
   Rectangle(float l, float b) {
      length = l;
      breadth = b;
   }
   void area() {
      cout << "Area of Rectangle = " << length * breadth << endl;
   }
};

// Derived class Triangle
class Triangle : public Shape {
   float base, height;
public:
   Triangle(float b, float h) {
      base = b;
      height = h;
   }
   void area() {
      cout << "Area of Triangle = " << 0.5 * base * height << endl;
   }
};

int main() {
   Shape* s;   // base class pointer
```

```
    Circle c(5);
    Rectangle r(4, 6);
    Triangle t(4, 3);

    s = &c;
    s->area();

    s = &r;
    s->area();

    s = &t;
    s->area();

    return 0;
}
```

Slip 11

Q.1) Create a class Person with data members name and age. Derive a class Student from Person that adds roll_no and marks. Display all information using a function.

```
#include <iostream>
#include <string>
using namespace std;
// Base class
class Person {
protected:
string name;
int age;
public:
void setPersonDetails(string n, int a) {
name = n;
age = a;
}
};
// Derived class
class Student : public Person {
private:
int roll_no;
float marks;
public:
void setStudentDetails(int r, float m) {
roll_no = r;
marks = m;
}
void displayDetails() {
cout << "Name : " << name << endl;
cout << "Age : " << age << endl;
cout << "Roll No. : " << roll_no << endl;
cout << "Marks : " << marks << endl;
}
};
// Main function
int main() {
Student s;
// Set details
s.setPersonDetails("Alice", 20);
s.setStudentDetails(101, 92.5);
// Display all details
cout << "----- Student Details -----" << endl;
s.displayDetails();
return 0;
}
```

Q.2) Write a C++ program to read Item information such as Itemno, Itemname , Itemprice, Quantity of 'n' Items. Write the Item information using file handling.

```
#include <iostream>
#include <fstream>
using namespace std;

class Item {
public:
    int itemNo;
    string itemName;
    float itemPrice;
    int quantity;

    void getData() {
        cout << "Enter Item No: ";
        cin >> itemNo;
        cout << "Enter Item Name: ";
        cin >> itemName;
        cout << "Enter Item Price: ";
        cin >> itemPrice;
        cout << "Enter Quantity: ";
        cin >> quantity;
    }

    void display() {
        cout << itemNo << "\t" << itemName << "\t" << itemPrice << "\t" << quantity << endl;
    }
};

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

    ofstream fout("items.txt");   // open file for writing

    Item item;
    for (int i = 0; i < n; i++) {
        cout << "\nEnter details of item " << i + 1 << ":\n";
        item.getData();
        fout << item.itemNo << " " << item.itemName << " "
             << item.itemPrice << " " << item.quantity << endl;
    }

    fout.close();
    cout << "\nItem information written to file items.txt successfully!\n";
    // Reading back from file
    ifstream fin("items.txt");
    cout << "\nReading data from file:\n";
    cout << "ItemNo\tName\tPrice\tQuantity\n";
    int no, qty;
    string name;
    float price;

    while (fin >> no >> name >> price >> qty) {
        cout << no << "\t" << name << "\t" << price << "\t" << qty << endl;
    }

    fin.close();
    return 0;
}
```

SLIP 12

Q.1) Write a C++ program to print area of circle, square using inline function.

```
#include <iostream>
using namespace std;
// Define constant for PI
const float PI = 3.14159;
// Inline function to calculate area of a circle
inline float areaof Circle(float radius) {
return PI * radius * radius;
}
// Inline function to calculate area of a square
inline float areaOfSquare(float side) {
return side * side;
}
int main() {
float radius, side;
// Input radius of the circle
cout << "Enter radius of the circle: ";
cin >> radius;
// Input side of the square
cout << "Enter side of the square: ";
cin >> side;
// Display the areas
cout << "\nArea of Circle = " << areaOfCircle(radius) << endl;
cout << "Area of Square = " << areaOfSquare(side) << endl;
return 0;
}
```

Q.2) Write a C++ program to create a class Date which contains three data members as dd, mm, and yyyy. Create and initialize the object by using parameterized constructor and display date in dd Mon-yyyy format. (Input: 19-12-2025 Output: 19-Dec-2025) Perform validation for month.

```
#include <iostream>
#include <string>
using namespace std;
class Date {
int dd, mm, yyyy;
string monthName;
// Array of month names
string months[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
"Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
public:
// Parameterized constructor
Date(int d, int m, int y) {
dd = d;
yyyy = y;
if (m >= 1 && m <= 12) {
mm = m;
monthName = months[m - 1];
} else {
mm = 0;
monthName = "Invalid";
cout << "Error: Invalid month (" << m << ")." << endl;
}
}
// Function to display date
void display() {
if (mm == 0)
cout << "Invalid date." << endl;
else
cout << dd << "-" << monthName << "-" << yyyy << endl;
}
};
int main() {
// Valid date
Date d1(19, 12, 2014);
cout << "Date 1: ";
d1.display();
// Another valid date
Date d2(5, 7, 2020);
cout << "Date 2: ";
d2.display();
// Invalid month case
Date d3(10, 15, 2022);
cout << "Date 3: ";
d3.display();
return 0;
}
```

SLIP 14

Q.1) Write a program to find sum of numbers between 1 to n using constructor where value of n will be passed to the constructor.

```
#include using namespace std; class Sum { int n, total; public: // Parameterized constructor Sum(in#include <iostream>
using namespace std;
class Sum {
int n, total;
public:
// Parameterized constructor
Sum(int num) {
n = num;
total = (n * (n + 1)) / 2;   // Formula for sum of 1..n
}
// Function to display result
void display() {
cout << "Sum of numbers from 1 to " << n << " is: " << total << endl;
}
};
int main() {
int num;
cout << "Enter value of n: ";
cin >> num;
// Object creation with n passed to constructor
Sum s(num);
s.display();
return 0;
}t num) { n = num; total = (n * (n + 1)) / 2; // Formula for sum of 1..n } // Function to display result void display() { cout #include <iostream>
using namespace std;
class Sum {
```

```
int n, total;
public:
// Parameterized constructor
Sum(int num) {
n = num;
total = (n * (n + 1)) / 2;   // Formula for sum of 1..n
}
// Function to display result
void display() {
cout << "Sum of numbers from 1 to " << n << " is: " << total << endl;
}
};
int main() {
int num;
cout << "Enter value of n: ";
cin >> num;
// Object creation with n passed to constructor
Sum s(num);
s.display();
return 0;
}<< "Sum of numbers from#include <iostream>
using namespace std;
class Sum {
int n, total;
public:
// Parameterized constructor
Sum(int num) {
n = num;
total = (n * (n + 1)) / 2;   // Formula for sum of 1..n
}
```

```
// Function to display result
void display() {
cout << "Sum of numbers from 1 to " << n << " is: " << total << endl;
}
};
int main() {
int num;
cout << "Enter value of n: ";
cin >> num;
// Object creation with n passed to constructor
Sum s(num);
s.display();
return 0;
} 1 to " << n << " is: " << total << e#include <iostream>
using namespace std;
class Sum {
int n, total;
public:
// Parameterized constructor
Sum(int num) {
n = num;
total = (n * (n + 1)) / 2;   // Formula for sum of 1..n
}
// Function to display result
void display() {
cout << "Sum of numbers from 1 to " << n << " is: " << total << endl;
}
};
int main() {
int num;
```

```
cout << "Enter value of n: ";
cin >> num;
// Object creation with n passed to constructor
Sum s(num);
s.display();
return 0;
}ndl; } }; int main() { int num#include <iostream>
using namespace std;
class Sum {
int n, total;
public:
// Parameterized constructor
Sum(int num) {
n = num;
total = (n * (n + 1)) / 2;   // Formula for sum of 1..n
}
// Function to display result
void display() {
cout << "Sum of numbers from 1 to " << n << " is: " << total << endl;
}
};
int main() {
int num;
cout << "Enter value of n: ";
cin >> num;
// Object creation with n passed to constructor
Sum s(num);
s.display();
return 0;
}; cout << "Enter value of n: ";
```

```
 cin >> num; // Object creation with n passed to constructor
 Sum s(num); s.display();
return 0; }
```

Q.2) Create 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).

```
#include <iostream>
#include <string>
using namespace std;

class College {
   int College_Id;
   string College_Name;
   int Establishment_Year;
   string University_Name;

public:
   void accept() {
      cout << "\nEnter College Id: ";
      cin >> College_Id;
      cin.ignore();
      cout << "Enter College Name: ";
      getline(cin, College_Name);
      cout << "Enter Establishment Year: ";
      cin >> Establishment_Year;
      cin.ignore();
      cout << "Enter University Name: ";
      getline(cin, University_Name);
   }
```

```
    void display() {
        cout << "\nCollege Id: " << College_Id
             << "\nCollege Name: " << College_Name
             << "\nEstablishment Year: " << Establishment_Year
             << "\nUniversity Name: " << University_Name << endl;
    }

    // Overloaded display function: filter by University
    void display(string uni) {
        if (University_Name == uni)
            display();
    }

    // Overloaded display function: filter by Establishment Year
    void display(int year) {
        if (Establishment_Year == year)
            display();
    }
};

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

    College c[50]; // can hold up to 50 colleges
    for (int i = 0; i < n; i++) {
        cout << "\nEnter details of College " << i + 1 << ":\n";
        c[i].accept();
    }

    int choice;
    do {
        cout << "\nMenu:\n1. Display colleges of specified University"
             << "\n2. Display colleges of specified Establishment Year"
             << "\n3. Exit\nEnter choice: ";
        cin >> choice;
        cin.ignore();

        if (choice == 1) {
            string uni;
            cout << "Enter University Name: ";
            getline(cin, uni);
            for (int i = 0; i < n; i++)
                c[i].display(uni);
        }
        else if (choice == 2) {
            int year;
            cout << "Enter Establishment Year: ";
            cin >> year;
            for (int i = 0; i < n; i++)
                c[i].display(year);
        }
    } while (choice != 3);

    return 0;
}
```

SLIP 15

Q.1) Write a C++ program to create an inline function to calculate the area of a rectangle with default value for width.

```
#include <iostream>
using namespace std;

// Inline function to calculate area of rectangle with default width

inline float areaOfRectangle(float length, float width = 5.0) {
   return length * width;
}

int main() {
   float length, width;

   // Input length

   cout << "Enter the length of the rectangle: ";
   cin >> length;

   // Ask user if they want to enter width or use default

   char choice;
   cout << "Do you want to enter width? (y/n): ";
   cin >> choice;

   if (choice == 'y' || choice == 'Y') {
      cout << "Enter the width of the rectangle: ";
      cin >> width;
      cout << "Area of Rectangle = " << areaOfRectangle(length, width) << endl;
   } else {
```

```
 cout << "Using default width = 5.0" << endl;
cout << "Area of Rectangle = " << areaOfRectangle(length) << endl;
}
return 0;
}
```

Q.2) 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.

```
#include <iostream>
#include <string>
using namespace std;

class Employee {
protected:
    string Name, Designation;
public:
    void acceptEmployee() {
        cin.ignore();
        cout << "Enter Employee Name: ";
        getline(cin, Name);
        cout << "Enter Designation: ";
        getline(cin, Designation);
    }
    void displayEmployee() {
        cout << "Name: " << Name << ", Designation: " << Designation;
    }
};
```

```
class Project {
protected:
    int Project_Id;
    string Title;
public:
    void acceptProject() {
        cout << "Enter Project Id: ";
        cin >> Project_Id;
        cin.ignore();
        cout << "Enter Project Title: ";
        getline(cin, Title);
    }
    void displayProject() {
        cout << ", Project Id: " << Project_Id << ", Title: " << Title;
    }
};

class Emp_Proj : public Employee, public Project {
    int Duration;
public:
    void accept() {
        acceptEmployee();
        acceptProject();
        cout << "Enter Project Duration (in months): ";
        cin >> Duration;
    }
    void display() {
        displayEmployee();
        displayProject();
        cout << ", Duration: " << Duration << " months" << endl;
    }
    int getDuration() { return Duration; }
};

int main() {
    Emp_Proj ep[50];
    int n = 0, choice;

    do {
        cout << "\nMenu:\n1. Build Master Table\n2. Display Master Table"
             << "\n3. Display Projects in Ascending Order of Duration\n4. Exit\nEnter choice: ";
        cin >> choice;

        if (choice == 1) {
            cout << "How many records? ";
            cin >> n;
            for (int i = 0; i < n; i++) {
                cout << "\nEnter details for record " << i + 1 << ":\n";
                ep[i].accept();
            }
        }
        else if (choice == 2) {
            cout << "\nMaster Table:\n";
            for (int i = 0; i < n; i++) {
                ep[i].display();
            }
        }
        else if (choice == 3) {
            // Simple bubble sort by Duration
            for (int i = 0; i < n - 1; i++) {
                for (int j = 0; j < n - i - 1; j++) {
                    if (ep[j].getDuration() > ep[j + 1].getDuration()) {
                        swap(ep[j], ep[j + 1]);
                    }
                }
            }
            cout << "\nProjects in Ascending Order of Duration:\n";
            for (int i = 0; i < n; i++) {
                ep[i].display();
            }
        }
    } while (choice != 4);

    return 0;
}
```

SLIP 17

Q.1) Write a C++ program to count the number of words in the given file.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
    ifstream file("input.txt");   // Open file (make sure input.txt exists in same folder)
    if (!file) {
        cout << "File not found!" << endl;
        return 0;
    }

    string word;
    int count = 0;

    while (file >> word) {   // Read word by word
        count++;
    }

    cout << "Total number of words in file: " << count << endl;
    file.close();
    return 0;
}
```

Q.2) 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 (Number_Of_Seats, 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 Business class.

```
#include <iostream>
#include <string>
using namespace std;
// Base class
class Flight {
protected:
int flight_no;
string flight_name;
public:
void setFlightDetails(int no, string name) {
flight_no = no;
flight_name = name;
}
void displayFlightDetails() const {
cout << "Flight No   : " << flight_no << endl;
cout << "Flight Name : " << flight_name << endl;
}
};
// Derived class from Flight
class Route : public Flight {
protected:
string source;
string destination;
```

```
public:
void setRouteDetails(string src, string dest) {
source = src;
destination = dest;
}
void displayRouteDetails() const {
cout << "Source
: " << source << endl;
cout << "Destination : " << destination << endl;
}
};
// Derived class from Route
class Reservation : public Route {
private:
int number_of_seats;
string travel_class;
float fare;
string travel_date;
public:
void setReservationDetails(int seats, string cls, float f, string date) {
number_of_seats = seats;
travel_class = cls;
fare = f;
travel_date = date;
}
void displayReservationDetails() const {
cout << "Seats
 : " << number_of_seats << endl;
cout << "Class
cout << "Fare
: " << travel_class << endl;
: $" << fare << endl;
cout << "Travel Date : " << travel_date << endl;
}
void displayAllDetails() const {
cout << "\n--- Reservation Details ---\n";
displayFlightDetails();
displayRouteDetails();
displayReservationDetails();
}
};
// Main function
int main() {
// Create object of Reservation
Reservation r;
// Set all details
r.setFlightDetails(1234, "Air India");
r.setRouteDetails("New Delhi", "Mumbai");
r.setReservationDetails(2, "Economy", 7500.50, "2025-09-20");
// Display all details
r.displayAllDetails();
return 0;
}
```

SLIP 22

Q.1) Write a C++ program to read a text file and count number of Uppercase Alphabets, Lowercase Alphabets, Digits and Spaces in it using File Handling.

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream file("input.txt");   // open file (make sure input.txt exists in same folder)
    if (!file) {
        cout << "File not found!" << endl;
        return 0;
    }

    char ch;
    int upper = 0, lower = 0, digit = 0, space = 0;

    while (file.get(ch)) {   // read character by character
        if (isupper(ch))
            upper++;
        else if (islower(ch))
            lower++;
        else if (isdigit(ch))
            digit++;
        else if (isspace(ch))
            space++;
    }

    cout << "Uppercase Letters: " << upper << endl;
    cout << "Lowercase Letters: " << lower << endl;
    cout << "Digits: " << digit << endl;
    cout << "Spaces: " << space << endl;

    file.close();
    return 0;
}
```

Q.2) Consider a class Point containing x and y coordinates. Write a C++ program to implement necessary functions to accept a point, to display a point and to find distance between two points using operator overloading (-). (Use friend function).

```
#include <iostream>
#include <cmath>
using namespace std;

class Point {
    float x, y;

public:
    void accept() {
        cout << "Enter x and y coordinates: ";
        cin >> x >> y;
    }

    void display() {
        cout << "(" << x << ", " << y << ")";
    }

    // Friend function to overload - operator
    friend float operator-(Point p1, Point p2);
};

// Operator overloading to calculate distance
float operator-(Point p1, Point p2) {
    return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}

int main() {
    Point p1, p2;

    cout << "Enter first point:\n";
    p1.accept();
    cout << "Enter second point:\n";
    p2.accept();

    cout << "\nFirst Point: ";
    p1.display();
    cout << "\nSecond Point: ";
    p2.display();

    float distance = p1 - p2;   // using overloaded operator
    cout << "\n\nDistance between points = " << distance << endl;

    return 0;
}
```

SLIP 23

Q.1) Write a C++ program using function to count and display the number of lines not starting with alphabet 'C' in a text file.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

// Function to count lines not starting with 'C'
int countLines(string filename) {
   ifstream file(filename);
   if (!file) {
      cout << "File not found!" << endl;
      return 0;
   }

   string line;
   int count = 0;

   while (getline(file, line)) {
      if (line.empty() || line[0] != 'C') {   // check first character
         count++;
         cout << line << endl;  // display such lines
      }
   }

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

```
int main() {
    string filename = "input.txt";  // file name
    int total = countLines(filename);

    cout << "\nNumber of lines not starting with 'C': " << total << endl;
    return 0;
}
```

Q.2) Write a program to design a class Complex to represent complex number. The Complex class should use an external function (use it 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.

```
#include <iostream>
using namespace std;

class Complex {
private:
    float real;
    float imag;

public:
    // Constructor to initialize values
    Complex() {
        real = 0;
        imag = 0;
    }

    // Function to accept complex number
    void input() {
        cout << "Enter real part: ";
        cin >> real;
        cout << "Enter imaginary part: ";
        cin >> imag;
    }

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

    // Declare friend function
    friend Complex addComplex(Complex c1, Complex c2);
};

// Friend function to add two complex numbers
Complex addComplex(Complex c1, Complex c2) {
    Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}

int main() {
    Complex num1, num2, sum;

    cout << "Enter first complex number:\n";
    num1.input();

    cout << "Enter second complex number:\n";
    num2.input();
sum = addComplex(num1, num2);
cout << "\nSum of complex numbers: ";
sum.display();
return 0;
}
```

SLIP 24

Q.1) 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.

```
#include <iostream>
using namespace std;
class Number {
int a, b;   // data members
public:
// Default constructor
Number() {
a = 0;
b = 0;
}
// Parameterized constructor
Number(int x, int y) {
a = x;
b = y;
}
// Member function to display maximum
void displayMax() {
if (a > b)
cout << "Maximum of (" << a << ", " << b << ") is: " << a << endl;
else if (b > a)
cout << "Maximum of (" << a << ", " << b << ") is: " << b << endl;
else
cout << "Both numbers (" << a << ", " << b << ") are equal." << endl;
}
};
// Main function
int main() {
// Object using default constructor
Number n1;
cout << "Object n1 (Default Constructor):" << endl;
n1.displayMax();
// Object using parameterized constructor
Number n2(25, 40);
cout << "\nObject n2 (Parameterized Constructor):" << endl;
n2.displayMax();
Number n3(15, 15);
cout << "\nObject n3 (Parameterized Constructor with equal numbers):" <<
endl;
n3.displayMax();
return 0;
}
```

Q.2) Create a class Employee and use a friend function to calculate the average salary from an array of employees.

```
#include <iostream>
using namespace std;

class Employee {
private:
   int id;
   string name;
```

```
    float salary;

public:
    // Constructor to initialize employee data
    Employee() {
        id = 0;
        name = "";
        salary = 0.0;
    }

    // Function to accept employee details
    void getData() {
        cout << "Enter ID: ";
        cin >> id;
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter Salary: ";
        cin >> salary;
    }

    // Friend function declaration
friend float calculateAverageSalary(Employee emp[], int size);
};
// Friend function to calculate average salary
float calculateAverageSalary(Employee emp[], int size) {
float total = 0;
for (int i = 0; i < size; i++) {
total += emp[i].salary;
}
return (size > 0) ? (total / size) : 0;
```

```
}
int main() {
int n;
cout << "Enter number of employees: ";
cin >> n;
Employee emp[100]; // assuming max 100 employees
for (int i = 0; i < n; i++) {
cout << "\nEnter details of employee " << i + 1 << ":\n";
emp[i].getData();
}
float avg = calculateAverageSalary(emp, n);
cout << "\nAverage Salary = " << avg << endl;
return 0;
}
```

SLIP 25

Q.1) Write a C++ program to create a class Employee having data members emp_id and emp_name and basic_salary. Accept this data for 5 variables and display the details of employee having salary > 5000.

```
#include <iostream>
using namespace std;

// Define a structure for Employee
struct Employee {
    int emp_id;
    char emp_name[50];
    float basic_salary;
};

int main() {
    Employee emp[5];

    // Accept data for 5 employees
    for (int i = 0; i < 5; i++) {
        cout << "\nEnter details for Employee " << i + 1 << ":\n";
        cout << "Enter Employee ID: ";
        cin >> emp[i].emp_id;
        cout << "Enter Employee Name: ";
        cin >> emp[i].emp_name;
        cout << "Enter Basic Salary: ";
        cin >> emp[i].basic_salary;
    }

    // Display employees with salary > 5000
    cout << "\nEmployees with salary more than 5000:\n";
    for (int i = 0; i < 5; i++) {
        if (emp[i].basic_salary > 5000) {
            cout << "\nEmployee ID: " << emp[i].emp_id << endl;
            cout << "Employee Name: " << emp[i].emp_name << endl;
            cout << "Basic Salary: " << emp[i].basic_salary << endl;
        }
    }

    return 0;
}
```

Q.2) Write a C++ program to create a class Student with data members roll_no, name and marks. Use a friend function to find and display the student with the highest marks among two students.

```
#include <iostream>
using namespace std;

class Student {
    int roll_no;
    float percentage;

public:
    // Function to accept data
    void accept() {
        cout << "Enter roll number: ";
        cin >> roll_no;
        cout << "Enter percentage: ";
        cin >> percentage;
    }

    // Function to return percentage
    float getPercentage() {
        return percentage;
    }

    // Function to return roll number
    int getRollNo() {
        return roll_no;
    }
};

int main() {
Student s1, s2;
cout << "Enter details for Student 1:\n";
s1.accept();
cout << "\nEnter details for Student 2:\n";
s2.accept();
cout << "\nStudent with higher percentage:\n";
if (s1.getPercentage() > s2.getPercentage()) {
cout << "Roll Number: " << s1.getRollNo() << endl;
} else if (s2.getPercentage() > s1.getPercentage()) {
cout << "Roll Number: " << s2.getRollNo() << endl;
} else {
cout << "Both students have equal percentage.\n";
}
return 0;
}
```