Task # 01

```cpp
#include <iostream>

int main()
{
    int num_1 = 1990;
    int num_2;
    for (int i=0; i<=3; i++) {
        if(i==0) {
            num_2 = 135;
        } else if(i==1) {
            num_2 = 7290;
        } else if(i==2) {
            num_2 = 11300;
        } else if(i==3) {
            num_2 = 16200;
        }

        std::cout << num_1 + i << " " << num_2 << "\n";
    }
    return 0;
}
```

Task # 02

```cpp
#include <iostream>

int main()
{
    std::cout << "Select Conversion:\n1.) Celsius to Fahrenheit??\n2.)
Fahrenheit to Celsius??\n";
    int choice;
    std::cin >> choice;
    int temperature;

    switch (choice)
    {
    case 1:
        std::cout << "Enter temperature: ";
        std::cin >> temperature;
        std::cout << "Temperature in Fahrenheit: " << (temperature * 9 / 5) +
32;
        break;
    case 2:
        std::cout << "Enter temperature: ";
        std::cin >> temperature;
        std::cout << "Temperature in Celsius: " << (temperature - 32) * 5/9;
```

```
        break;
    default:
        break;
    }
    return 0;
}
```

Task # 03

```
#include <iostream>

int main()
{
    std::cout << "Curreny Converter!!\n";
    std::cout << "Enter amount in $: ";

    float usd = 1.487;
    float french = 0.172; // Actual Value is 0.93
    float euro = 0.584; // Actual Value is 0.93
    float yen = 0.00955; // Actual Value is 147.77
    float input;
    std::cin >> input;

    std::cout << "$" << input << " in USD: " << input << "\n$" << input << "
in French: " << input * french << "\n$" << input << " in German Euro: " <<
input * euro << "\n$" << input << " in Yen: " << input*yen;

    return 0;
}
```

Task # 04

```
#include <iostream>

struct fraction
{
    int num;
    int den;
};


int main()
{
   fraction f_frac;
   fraction s_frac;
   char c;
```

```
   std::cout << "Enter first Fraction (x/y): ";
   std::cin >> f_frac.num >> c >> f_frac.den;

   std::cout << "Enter second Fraction (x/y): ";
   std::cin >> s_frac.num >> c >> s_frac.den;

   std::cout << "Output: " << ((f_frac.num*s_frac.den) +
(s_frac.num*f_frac.den)) << "/" << f_frac.den*s_frac.den;
    return 0;
}
```

Task # 05

```
#include <iostream>

int main()
{
    char vowel;
    char *ptr = &vowel;

    std::cout << "Enter a Letter: ";
    std::cin >> vowel;

    if(*ptr == 'a' || 'A') {
        std::cout << "Vowel";
    } else if(*ptr == 'e' || 'E') {
        std::cout << "Vowel";
    } else if(*ptr == 'i' || 'I') {
        std::cout << "Vowel";
    } else if(*ptr == 'o' || 'O') {
        std::cout << "Vowel";
    } else if(*ptr == 'U' || 'u') {
        std::cout << "Vowel";
    } else {
        std::cout << "Not a Vowel";
    }

    return 0;
}
```

Task # 06

```cpp
#include <iostream>

int main()
{
   int a, b, c;

   std::cout << "Enter side a of Triangle: ";
   std::cin >> a;
   std::cout << "Enter side b of Triangle: ";
   std::cin >> b;
   std::cout << "Enter side c of Triangle: ";
   std::cin >> c;

   if((a + b) > c) {
    if((c + b) > a) {
        std::cout << "Two Sides are greater!";
    } else if((a + c) > b) {
        std::cout << "Two Sides are greater!";
    } else {
        std::cout << "Not a triangle inequality theorem!";
    }
} else if((c + b) > a) {
    if((a + b) > c) {
        std::cout << "Two Sides are greater!";
    } else if((a + c) > b) {
        std::cout << "Two Sides are greater!";
    } else {
        std::cout << "Not a triangle inequality theorem!";
    }
} else if((a + c) > b) {
    if((a + b) > c) {
        std::cout << "Two Sides are greater!";
    } else if((c + b) > a) {
        std::cout << "Two Sides are greater!";
    } else {
        std::cout << "Not a triangle inequality theorem!";
    }
} else {
    std::cout << "Not a triangle inequality theorem!";
}
    return 0;
}
```

Task # 01

```cpp
#include <iostream>

int main()
{
    int input, ans, count = 1;

    std::cout << "Enter a Number: ";
    std::cin >> input;

    for (int i=1; i<=20; i++) {
        for (int j=1; j<=10; j++) {
            ans = input *count;
            if(i<=3) {
            std::cout << ans << "\t";
        }
            count ++;
        }
        std::cout << "\n";
    }
    return 0;
}
```

Task # 02

```cpp
#include <iostream>

int main()
{
    std::cout << "Select Conversion:\n1.) Celsius to Fahrenheit??\n2.) Fahrenheit to Celsius??\n";
    int choice;
    std::cin >> choice;
    float temperature;

    switch (choice)
    {
    case 1:
        std::cout << "Enter temperature: ";
        std::cin >> temperature;
        std::cout << "Temperature in Fahrenheit: " << (temperature * 9 / 5) + 32;
        break;
    case 2:
        std::cout << "Enter temperature: ";
        std::cin >> temperature;
        std::cout << "Temperature in Celsius: " << (temperature - 32) * 5 / 9;
        break;

    default:
        break;
    }
    return 0;
}
```

Task # 03

```cpp
#include <iostream>

struct calculator
{
    float f_num;
    float s_num;
};

int main()
{
    calculator ans;
    char oper;
    std::cout << "Enter first number, operator, second number (10 / 3): ";
    std::cin >> ans.f_num >> oper >> ans.s_num;

    switch (oper)
    {
    case '+':
        std::cout << "Answer = " << ans.f_num + ans.s_num;
        break;
    case '-':
        std::cout << "Answer = " << ans.f_num - ans.s_num;
        break;
    case '*':
        std::cout << "Answer = " << ans.f_num * ans.s_num;
        break;
    case '/':
        std::cout << "Answer = " << ans.f_num / ans.s_num;
        break;

    default:
        break;
    }
    return 0;
}
```

Task # 04

```cpp
#include <iostream>

int main()
{
    for (int i=0; i<20; i++) {
        for (int j=0; j<=20-i; j++) {
            std::cout << " ";
        }
        for (int k=1; k<=2*i-1; k++) {
            std::cout << "X";
        }
        std::cout << std::endl;
    }
    return 0;
}
```

Task # 05

```cpp
#include <iostream>

using namespace std;
int main() {
    float initial_amount;
    int years;
    float percent;
    // float output;

    cout << "Enter initial amount: ";
    cin >> initial_amount;

    cout << "Enter number of years: ";
    cin >> years;

    cout << "Enter interest rate (percent per year): ";
    cin >> percent;
    percent /= 100;

    // output = initial_amount;
    for (int i=0; i<2; i++) {
        // output = output + (output*percent);
        initial_amount = initial_amount + (initial_amount*percent);

    }
    cout << initial_amount;



    return 0;
}
```

Task # 06

```cpp
#include <iostream>
using namespace std;
int main() {
    int guests = 6;
    int track = guests;
    for (int i=0; i<3; i++) {
        guests *=--track;
    }
    cout << "Guests: " << guests;

    return 0;
}
```

Task # 07

```cpp
#include <iostream>

using namespace std;

void reverseNum1(int num) {
    int u, t, h;
    u= num % 10;
    t= (num / 10) % 10;
    h= (num / 100) % 10;

    cout << u << t << h << endl;
}

void reverseNum2(int num) {
    while (num>0)
    {
        cout << num%10; // This is for to get the last digit.
        num/=10; // This is for to remove the last digit
    }

}

int main() {
    reverseNum1(123); // 321
    reverseNum2(12345); // 54321
    return 0;
}
```

Task # 08

```cpp
#include <iostream>

int main() {
    int arr[] = {10, 20, 4, 45, 99, 55};
    int largest=0;
    int secLargest=0;
    for (int i=0; i< 6; i++) {
        if(arr[i] > largest) {
            secLargest = largest;
            largest = arr[i];
        } else if(arr[i] < largest && arr[i] > secLargest) {
            secLargest = arr[i];
        }
    }

    std::cout << secLargest << std::endl;
    return 0;
}
```

Task # 01:

```cpp
#include <iostream>

using namespace std;

struct PhoneNumber {
    int c;
    int e;
    int n;
};

int main() {
    PhoneNumber p;
    cout << "Enter your area code, exchange, and number: "; cin >> p.c >> p.e
>> p.n;
    cout << "My number is (212) 767-8900" << endl;
    cout << "Your number is (" << p.c << ") " << p.e << "-" << p.n;

    return 0;
}
```

Task # 02:

```cpp
#include <iostream>

using namespace std;

struct Employee {
    int num;
    float com;
};

void displayInfo(Employee employees[], int size);

int main() {
    const int EMP_COUNT = 3;
    Employee employees[EMP_COUNT];

    cout << "Fill the data for 3 employees:\n";

    for (int i = 0; i < EMP_COUNT; i++) {
        cout << "Employee " << i + 1 << " (number) (compensation): ";
        cin >> employees[i].num >> employees[i].com;
    }

    displayInfo(employees, EMP_COUNT);
    return 0;
}

void displayInfo(Employee employees[], int size) {
    cout << "\nEmployee Details:\n";
    for (int i = 0; i < size; i++) {
        cout << "Employee " << i + 1 << " - Number: " << employees[i].num
             << ", Compensation: $" << employees[i].com << endl;
    }
}
```

Task # 03:

```cpp
#include <iostream>

using namespace std;

struct Time {
    int h;
    int m;
    int s;
};

int main() {
    Time t;
    long totalSecs;
    cout << "Enter Time in (hours:minutes:seconds): "; cin >> t.h >> t.m >>
t.s;
    totalSecs = t.h*3600 + t.m*60 + t.s;

    cout << "Total Seconds: " << totalSecs;

    return 0;
}
```

Task # 04:

```cpp
#include <iostream>

using namespace std;

int main() {
    int len = 5;
    float arr[len], avg=0;

    for (int i=0; i<len; i++) {
        cout << "Enter number " << i+1 << ": ";
        cin >> *(arr + i);
        /*
        arr -> itself &arr[0]
        1. // 100 + (0*4)
        2. // 100 + (1*4)
        */
       avg += *(arr + i);
    }
    cout << "\nAverage = " << avg/len;

    return 0;
}
```

Task # 05:

```cpp
#include <iostream>

using namespace std;

int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    int *ptr = arr; // arr -> &arr[0]

    for (int i=0; i<5; i++) {
        cout << "Number " << i+1 << ": " << *ptr << endl;
        ptr++; // Shifting to the next address
    }

    return 0;
}
```

Task # 06:

```cpp
#include <iostream>

using namespace std;

struct Distance {
    int feet;
    float inches;
};

struct Room {
    Distance length;
    Distance width;
};

int main() {
    Room r;
    r.length.feet = 12;
    r.length.inches = 10.0;

    r.width.feet = 10;
    r.width.inches = 8.0;

    float totalLen, totalW;
    totalLen = r.length.feet + (r.length.inches/12.0);
    totalW = r.width.feet + (r.width.inches/12.0);

    cout << "Area in Sq: " << totalLen * totalW;

    return 0;
}
```

Task 01:

```cpp
#include <iostream>
#include <conio.h>
using namespace std;

class toolBooth {
private:
    unsigned int cars;
    double amount;
public:
    toolBooth() : cars(0), amount(0) {}
    void payingCar() {
        cars++;
        amount+=0.50;
    }

    void nopayCar() {
        cars++;
    }

    void display() {
        cout << "Cars: " << cars << endl;
        cout << "Amount: " << amount << endl;
    }
};

int main() {
    toolBooth t1;
    char ch;
    cout << "Press 'P' for a paying car, 'N' for a non-paying car, and ESC to
exit.\n";

    while (true) {
        ch = _getch();  // capture keypress

        if (ch == 27) {  // esc key (ASCII 27) to exit
            t1.display();
            break;
        } else if (ch == 'p' || ch == 'P') {
            t1.payingCar();
            cout << "Paying car counted!\n";
        } else if (ch == 'n' || ch == 'N') {
            t1.nopayCar();
            cout << "Non-paying car counted!\n";
        }
    }
    return 0;
}
```

Task 02:

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

class time {
private:
    int hours;
    int minutes;
    int seconds;
public:
    time() : hours(0), minutes(0), seconds(0) {} // Default
    time(int hrs, int min, int sec) : hours(hrs), minutes(min), seconds(sec)
{} // Parameterized

    void display() const {
        cout << "Time: " << hours << ":" << minutes << ":" << seconds << endl;
    }
    void addTime(time, time);
};

void time::addTime(time t1, time t2) {
    hours = t1.hours + t2.hours;
    minutes = t1.minutes + t2.minutes;
    seconds = t1.seconds + t2.seconds;

    if (seconds >= 60) {
        minutes += seconds / 60;
        seconds %= 60;
    }
    if (minutes >= 60) {
        hours += minutes / 60;
        minutes %= 60;
    }
    hours %= 24; // it's bcz tracking hours in 24-format cause the days aren't
mention in the
}

int main() {
    time t1(12, 20, 40), t2(24, 10, 25), t3(6, 3, 60);
    t1.display();
    t2.display();
    t3.display();
    cout << "=================" << endl;
    t1.addTime(t2, t3);
    t1.display();

    return 0;
}
```

Task 03:

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

class Employee {
private:
    int empNumber;
    float compensation;

public:
    void getData() {
        cout << "Enter Employee Number: ";
        cin >> empNumber;
        cout << "Enter Compensation: ";
        cin >> compensation;
    }

    void display() const {
        cout << "Employee Number: " << empNumber << " | Compensation: $";
        int dollars = compensation;
        int cents = (int)((compensation - dollars) * 100);
        cout << dollars << "." << ((cents) < 10 ? "0" : "") << cents << endl;
    }
};

int main() {
    const int numEmployees = 3;
    Employee employees[numEmployees];

    for (int i = 0; i < numEmployees; i++) {
        cout << "Enter details for Employee " << i + 1 << ":\n";
        employees[i].getData();
    }

    cout << "\nEmployee Details:\n";
    for (int i = 0; i < numEmployees; i++) {
        employees[i].display();
    }

    return 0;
}
```

Task 04:

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

class BankAccount {
private:
    int accnumber;
    float balance;
public:
    BankAccount() : accnumber(0), balance(0) {}
    BankAccount(int acc, float b) : accnumber(acc), balance(b) {}
    void withDrawal(float);
    void deposit(float);
    void transfer(BankAccount&, float);
};
void BankAccount::withDrawal(float amount) {
    if(amount > this->balance) cout << "Insufficient Balance!!" << endl;
    else {
    balance -= amount;
    cout << "Amount: $" << amount << " withdrawal successfully!!";
    }
}
void BankAccount::deposit(float amount) {
    balance += amount;
    cout << "Amount: $" << amount << " deposited successfully to Account " <<
this->accnumber << "!!" << endl;
}
void BankAccount::transfer(BankAccount& b, float a) {
    if(this->balance < a) cout << "Insufficient Balance!!" << endl;
    else {
    this->balance -= a;
    b.balance += a;
    cout << "\nSuccessfully Transfer!!\nAccount Number: " << this->accnumber
<< "\nTransfered Account Number: " << b.accnumber << "\nYour Balance: $" <<
this->balance << "\nReciever Balance: $" << b.balance << endl;
    }
}
int main() {
    cout << "==== Welcome ====" << endl;
    BankAccount b1(1, 200), b2(2, 900), b3(3, 100.33);
    b1.deposit(32.99);
    b2.deposit(32.99);
    b3.deposit(32.99);

    b1.transfer(b2, 30.0);

    return 0;
}
```

Task 05:

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

class Distance {
private:
    int feet, inches;
public:
    Distance() : feet(0), inches(0) {}
    Distance(int f, int i) : feet(f), inches(i) {}

    Distance ADD(const Distance&);
    void display() const;
};

Distance Distance::ADD(const Distance& d) {
    Distance result;
    result.feet = feet + d.feet;
    result.inches = inches + d.inches;

    return result;
}

void Distance::display() const {
    cout << "Total Distance: " << feet << " feet " << inches << " inches." <<
endl;
}

int main() {
    Distance d1(20, 30), d2(10, 80);
    Distance d3 = d1.ADD(d2);
    d3.display();
    return 0;
}
```

Task 06:

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

class Angle {
private:
    int degrees;
    float minutes;
    char direction;

public:
    void getInput() {
        cout << "Enter degrees: ";
        cin >> degrees;
        cout << "Enter minutes: ";
        cin >> minutes;
        cout << "Enter direction (N/S/E/W): ";
        cin >> direction;
    }

    void display() const {
        cout << degrees << "°" << minutes << "' " << direction;
    }
};

class Ship {
private:
    static int counter;
    int serialNumber;
    Angle latitude, longitude;

public:
    Ship() {
        serialNumber = ++counter;
    }

    void setLocation() {
        cout << "Enter latitude for Ship " << serialNumber << ":\n";
        latitude.getInput();
        cout << "Enter longitude for Ship " << serialNumber << ":\n";
        longitude.getInput();
    }

    void display() const {
        cout << "Ship Serial Number: " << serialNumber << "\n";
        cout << "Latitude: "; latitude.display();
        cout << "\nLongitude: "; longitude.display();
        cout << "\n----------------------\n";
```

```
    }
};

int Ship::counter = 0;

int main() {
    const int numShips = 3;
    Ship ships[numShips];

    for (int i = 0; i < numShips; i++) {
        ships[i].setLocation();
    }

    cout << "\nShip Details:\n";
    for (int i = 0; i < numShips; i++) {
        ships[i].display();
    }

    return 0;
}
```

Lab 01:

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

using namespace std;
class Book {
private:
    string title;
    string author;
    double price;
    static double discountRate;

public:
    Book(string t, string a, double p) : title(t), author(a), price(p) {}

    void setPrice(double p) { price = p; }

    static void setDiscountRate(double d) { discountRate = d; }

    void display() const {
        double discountedPrice = price - (price * (discountRate / 100));
        cout << "Title: " << title
             << "\nAuthor: " << author
             << "\nOriginal Price: $" << price
             << "\nDiscounted Price: $" << discountedPrice << "\n" << endl;
    }
};

double Book::discountRate = 0;

int main() {
    Book::setDiscountRate(20);

    Book b1("OOPS in C++", "ABC", 200);
    Book b2("Data Structures", "XYZ", 300);

    b1.display();
    b2.display();

    return 0;
}
```

Lab 02:

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

class Appliance {
private:
    string name;
    int rating;
    int time;

public:

    void setName(string n) { name = n; }
    void setRating(int r) { rating = r; }
    void setTime(int t) { time = t; }


    string getName() { return name; }
    int getRating() { return rating; }
    int getTime() { return time; }


    int dailyEnergyConsumption() {
        return rating * time;
    }


    static void compare(Appliance a1, Appliance a2) {
        int energyA1 = a1.dailyEnergyConsumption();
        int energyA2 = a2.dailyEnergyConsumption();

        if (energyA1 > energyA2) {
            cout << a1.getName() << " consumes more energy daily!" << endl;
        } else if (energyA2 > energyA1) {
            cout << a2.getName() << " consumes more energy daily!" << endl;
        } else {
            cout << "Both appliances consume equal energy daily!" << endl;
        }
    }
};

int main() {
    Appliance a1, a2;


    a1.setName("Refrigerator");
    a1.setRating(2000);
    a1.setTime(5);


    a2.setName("Washing Machine");
    a2.setRating(1500);
    a2.setTime(2);

    Appliance::compare(a1, a2);

    return 0;
}
```

Lab 03:

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

class Course {
private:
    string name;
    int course_id;
    float fee;
    static int totalCourses;

public:
    Course(string name, int id, float fee) { this->name = name; this->course_id = id; this->fee
= fee; totalCourses++;}

    void setName(string n) {
        this->name = n;
    }

    string getName() {
        return this->name;
    }

    void displayCourse() {
        cout << "Course: " << this->name << endl;
    }

    static int getTotalCourses() {
        return totalCourses;
    }
};

int Course::totalCourses = 0;

int main() {
    Course c1("OOPS", 1, 200.0), c2("DSA", 2, 300), c3("Algorithms", 3, 400.0);
    cout << "Total No. of Courses: " << Course::getTotalCourses() << endl;
    return 0;
}
```

Lab 04:

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

class Stock
{
private:
    string companyName;
    float stockPrice;
    int availableShares;
    static float marketValue;

public:
    Stock(string name, float price, int shares)
    {
        this->companyName = name;
        this->stockPrice = price;
        this->availableShares = shares;
    }

    void market_Value()
    {
        marketValue = stockPrice * availableShares;
    }

    static float getMarketValue() {
        return marketValue;
    }

    void displayStock()
    {
        cout << "📈 " << companyName
             << " | Price: $" << stockPrice
             << " | Shares: " << availableShares << endl;
    }

    void setPrice(float p) { this->stockPrice = p; }

    void buyShares(int quantity)
    {
        if (quantity > availableShares)
        {
            cout << "Not enough shares available to buy." << endl;
            return; // Stop
        }

        availableShares -= quantity;

        if (quantity > 10)
            stockPrice *= 1.01; // Increment of 1%

        market_Value();
        cout << "✅ Bought: " << quantity << " shares of " << companyName << endl;
    }

    void sellShares(int quantity)
    {
        availableShares += quantity;

        if (quantity > 10)
```

```
            stockPrice *= 0.99; // Decrement of 1%

        market_Value();
        cout << "✅ Sold: " << quantity << " shares of " << companyName << endl;
    }
};

float Stock::marketValue = 0.0;

int main()
{
    Stock apple("Apple", 150.0, 100),
    tesla("Tesla", 200.0, 80);


    apple.displayStock();
    tesla.displayStock();
    cout << "🌍 Total Market Value: $" << Stock::getMarketValue() << "\n\n";

    // Buy Shares
    apple.buyShares(15);
    tesla.buyShares(10);

    cout << "\n📊 After transactions:\n";
    apple.displayStock();
    tesla.displayStock();
    cout << "🌍 Total Market Value: $" << Stock::getMarketValue() << "\n";

    return 0;
}
```

Lab 05:

```
#include <iostream>
#include <cstdlib> // rand()
using namespace std;

class Ticket
{
private:
    int ticketNumber;
    string passengerName;
    int seatNumber;
    static int soldTickets;

public:
    Ticket(string pN, int sN)
    {
        ticketNumber = rand() % 1000 + 1;
        passengerName = pN;
        seatNumber = sN;
        soldTickets++;
    }

    void display() {
        cout << "Ticket No: " << ticketNumber << ", Passenger: " << passengerName << ", Seat: "
<< seatNumber << endl;
    }

    static int getSoldTickets() {
```

```
        return soldTickets;
    }
};

int Ticket::soldTickets = 0;

int main() {

    Ticket t1("Alice", 12);
    Ticket t2("Bob", 15);

    t1.display();
    t2.display();

    cout << "Total Tickets Sold: " << Ticket::getSoldTickets() << endl;

    return 0;
}
```

Task # 01

```cpp
#include <iostream>
using namespace std;
class Rectangle
{
private:
    int width;
    int length;
    int area;

public:
    Rectangle() : width(0), length(0), area(0) {}
    Rectangle(int a, int b) : width(a), length(b), area(a * b) {}
    Rectangle(Rectangle &c) : width(c.width), length(c.length), area(c.area)
{}
    void display()
    {
        cout << "Area of Rectangle is " << area << endl;
    }
    void setvalue(int a, int b)
    {
        width = a;
        length = b;
        area = a * b;
    }
    Rectangle operator+(const Rectangle &c)
    {
        Rectangle temp;
        temp.width = width + c.width;
        temp.length = length + c.length;
        temp.area = area + c.area;
        return temp;
    }
};
int main()
{
    Rectangle c1(2, 5);
    Rectangle c2(4, 6);
    c1.display();
    c2.display();
    Rectangle c3 = c1 + c2;
    c3.display();
    return 0;
}
```

Task # 02

```cpp
#include <iostream>
using namespace std;
class Distance
{
private:
    int feet;
    float inch;

public:
    Distance() : feet(0), inch(0) {}
    Distance(int a, float b) : feet(a), inch(b) {}
    Distance(Distance &c) : feet(c.feet), inch(c.inch) {}
    void display()
    {
        cout << "Distance is " << feet << "\'" << inch << "\"" << endl;
    }
    void setvalue()
    {
        int a;
        float b;
        cout << "Enter feet: ";
        cin >> a;
        cout << "Enter inches: ";
        cin >> b;
        feet = a;
        inch = b;
    }
    void operator+=(const Distance &c)
    {
        feet += c.feet;
        inch += c.inch;
        if (inch > 12)
        {
            inch -= 12;
            feet++;
        }
    }
};
int main()
{
    Distance c1(8, 5.9);
    Distance c2;
    c2.setvalue();
    c1.display();
    c2.display();
    c2 += c1;
    cout << "Added ";
```

```
    c2.display();
    return 0;
}
```

Task # 03

```cpp
#include <iostream>
#include <string>
using namespace std;
class remarks
{
private:
    string remark;

public:
    remarks() : remark(" ") {}
    remarks(const string &example) : remark(example) {}
    void display()
    {
        cout << remark;
    }
    void setter(string h)
    {
        remark = h;
    }
    remarks operator+=(const remarks &ex)
    {
        return remarks(remark + ex.remark);
    }
};
int main()
{
    remarks c1;
    remarks c2("Need some improvement in Physics.");
    remarks finalremarks;
    c1.setter("Excellent in Mathematics.");
    cout << "First Remarks: ";
    c1.display();
    cout << endl
         << "Second Remarks: ";
    c2.display();
    finalremarks = c1 += c2;
    cout << endl
         << "Final Remarks: ";
    finalremarks.display();
    return 0;
}
```

Task # 04

```cpp
#include <iostream>
#include <string>
using namespace std;
class student
{
private:
    int marks[5];

public:
    student()
    {
        for (int i = 0; i < 5; i++)
            marks[i] = 0;
    }
    int &operator[](int index)
    {
        if (index >= 0 && index < 5)
            return marks[index];
        else
            cout << "Index out of boundary!";
    }
    void display()
    {
        int total = 0;
        for (int i = 0; i < 5; i++)
            total += marks[i];
        float avg = total / 5.00;
        cout << "Total marks: " << total << endl;
        cout << "Average marks: " << avg;
    }
};
int main()
{
    student s1;
    for (int i = 0; i < 5; i++)
    {
        cout << "Enter marks for subject" << i + 1 << ": ";
        cin >> s1[i];
    }
    s1.display();
    return 0;
}
```

Task # 05

```
#include <iostream>
#include <string>
using namespace std;
class YearTemperature
{
private:
    int temp[12];

public:
    YearTemperature()
    {
        for (int i = 0; i < 12; i++)
            temp[i] = 0;
    }
    int &operator[](int index)
    {
        if (index >= 0 && index < 12)
            return temp[index];
        else
            cout << "Index out of boundary!";
    }
    void display()
    {
        string months[12] = {"jan", "feb", "mar", "april", "may", "june",
"july", "aug", "sept", "oct", "nov", "dec"};
        for (int i = 0; i < 12; i++)
        {
            cout << months[i] << ":" << temp[i] << "\'C" << endl;
        }
    }
};
int main()
{
    YearTemperature y1;
    for (int i = 0; i < 12; i++)
    {
        cout << "Enter temperature for month " << i + 1 << ": ";
        cin >> y1[i];
    }
    cout << "All monthly temperatures: " << endl;
    y1.display();
    return 0;
}
```

Lab # 01

```cpp
#include <iostream>
#include <string>

using namespace std;

class Publication
{
    string title;
    float price;

public:
    Publication() : title("null"), price(0.0f) {};
    void getdata()
    {
        cout << "enter title: ";
        cin >> title;
        cout << "enter price: ";
        cin >> price;
    }
    void putdata()
    {
        cout << "title: " << title << endl;
        cout << "price: " << price << endl;
    }
};

class book : public Publication
{
    int pagecount;

public:
    book() : pagecount(0) {};
    void getdata()
    {
        Publication ::getdata();
        cout << "enter pagecount: ";
        cin >> pagecount;
    }
    void putdata()
    {
        Publication ::putdata();
        cout << "pagecount: " << pagecount << endl;
    }
};

class tape : public Publication
{
```

```
    float minutes;

public:
    tape() : minutes(0.0f) {};
    void getdata()
    {
        Publication ::getdata();
        cout << "enter minutes: ";
        cin >> minutes;
    }
    void putdata()
    {
        Publication ::putdata();
        cout << "minutes: " << minutes << endl;
    }
};

int main()
{
    book b1;
    tape t1;
    b1.getdata();
    t1.getdata();
    b1.putdata();
    t1.putdata();
}
```

Lab # 02

```
#include <iostream>
#include <string>

using namespace std;

class Sales{
    protected:
    float array[3];
    public:
    void getdata(){
        for(int i = 0; i<=2; i++){
            cout << "enter sales amount " << i + 1 <<":";
            cin >> array[i];
        }
    }
    void putdata(){
       for(int i=0; i<=2; i++){
        cout << "sales amount " << i+1 <<":"<< array[i] << endl;
       }
    }
};
class Publication{
    protected:
    string title;
    float price;

    public:
    Publication() : title("null"), price(0.0f) {};
    void getdata() {
       cout << "enter title: ";
       cin >> title;
       cout << "enter price: ";
       cin >> price;
    }
    void putdata() {
       cout << "title: " << title << endl;
       cout << "price: " << price<< endl;
    }
};

class book : public Publication, public Sales{
    protected:
    int pagecount;
    public:
    book() : pagecount(0) {};
    void getdata() {
        Sales ::getdata();
```

```
        Publication ::getdata();
        cout<< "enter pagecount: ";
        cin >> pagecount;
    }
    void putdata() {
        Sales ::putdata();
        Publication ::putdata();
        cout << "pagecount: " << pagecount << endl;
    }

};

class tape : public Publication, public Sales{
    protected:
    float minutes;
    public:
    tape() : minutes(0.0f) {};
    void getdata() {
        Sales ::getdata();
        Publication ::getdata();
        cout << "enter minutes: ";
        cin >> minutes;
    }
    void putdata() {
        Sales ::putdata();
        Publication ::putdata();
        cout << "minutes: " << minutes << endl;
    }
};

int main(){
    book b1;
    tape t1;
    b1.getdata();
    t1.getdata();
    b1.putdata();
    t1.putdata();
}
```

Lab # 03

```cpp
#include <iostream>
#include <string>

using namespace std;

class Publication{
    protected:
    string title;
    float price;

    public:
    Publication() : title("null"), price(0.0f) {};
    void getdata() {
       cout << "enter title: ";
       cin >> title;
       cout << "enter price: ";
       cin >> price;
    }
    void putdata() {
       cout << "title: " << title << endl;
       cout << "price: " << price<< endl;
    }
};

class Disk : public Publication{
    enum disktype {CD, DVD};
    disktype type;

    public:
    void getdata() {
        Publication ::getdata();
        char choice;
        cout << "enter your type: ";
        cin >> choice;
        if(choice == 'C' || choice == 'c')
          type = CD;
        else if(choice == 'D' || choice == 'd')
          type = DVD;
        else
          cout << "invalid choice.";
    }
    void putdata() {
        Publication ::putdata();
        cout << "Type: " << (type == CD ? "CD" : "DVD");
    }
};
```

Lab # 04

```cpp
#include <iostream>
#include <string>

using namespace std;

class Sales{
    protected:
    float array[3];
    public:
    void getdata(){
        for(int i = 0; i<=2; i++){
            cout << "enter sales amount " << i + 1 <<":";
            cin >> array[i];
        }
    }
    void putdata(){
       for(int i=0; i<=2; i++){
        cout << "sales amount " << i+1 <<":"<< array[i] << endl;
       }
    }
};
class Publication{
    protected:
    string title;
    float price;

    public:
    Publication() : title("null"), price(0.0f) {};
    void getdata() {
       cout << "enter title: ";
       cin >> title;
       cout << "enter price: ";
       cin >> price;
    }
    void putdata() {
       cout << "title: " << title << endl;
       cout << "price: " << price<< endl;
    }
};
class date{
    protected:
    int datee, month, year;
};

class Publication2: public Publication, public date{
    public:
    void getdata(){
```

```
        Publication ::getdata();
        cout << "enter date: ";
        cin >> datee;
        cout << "enter month: ";
        cin >> month;
        cout << "year: ";
        cin >> year;
    }
    void putdata(){
        Publication ::putdata();
        cout << "date: " << " " << datee << " - " << month << " - " << year <<
endl;
    }
};

class book : public Publication2, public Sales{
    protected:
    int pagecount;
    public:
    book() : pagecount(0) {};
    void getdata() {
        Publication2 ::getdata();
        Sales ::getdata();
        cout<< "enter pagecount: ";
        cin >> pagecount;
    }
    void putdata() {
        Publication2 ::putdata();
        Sales ::putdata();
        cout << "pagecount: " << pagecount << endl;
    }

};

class tape : public Publication2, public Sales{
    protected:
    float minutes;
    public:
    tape() : minutes(0.0f) {};
    void getdata() {
        Publication2 ::getdata();
        Sales ::getdata();
        cout << "enter minutes: ";
        cin >> minutes;
    }
    void putdata() {

        Publication2 ::putdata();
```

```
        Sales ::putdata();
        cout << "minutes: " << minutes << endl;
    }
};
class Disk : public Publication{
    enum disktype {CD, DVD};
    disktype type;

    public:
    void getdata() {
        Publication ::getdata();
        char choice;
        cout << "enter your type: ";
        cin >> choice;
        if(choice == 'C' || choice == 'c')
          type = CD;
        else if(choice == 'D' || choice == 'd')
          type = DVD;
        else
          cout << "invalid choice.";
    }
    void putdata() {
        Publication ::putdata();
        cout << "Type: " << (type == CD ? "CD" : "DVD");
    }
};
```

Lab # 05

```cpp
#include <iostream>
#include <string>

using namespace std;

class counterclass{
    protected:
    int count;
    public:
    counterclass() : count(0) {};
    int operator ++(){
        return ++count;
    }
    int operator --(){
        return --count;
    }
    void print(){
        cout << "count: " << count << endl;
    }

};
class post : public counterclass{
    public:
    int operator++(int){
        return count++;
    }
    int operator--(int){
        return count--;
    }
};
int main(){
    counterclass c1;
    ++c1;
    c1.print();
    post p1;
    p1++;
    p1.print();

}
```

Lab # 01

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

class Author {
public:
    string name;
    int birthYear;
    Author(string n, int y) : name(n), birthYear(y) {}
};

class Edition {
public:
    string language;
    int pubYear;
    Edition(string lang, int y) : language(lang), pubYear(y) {}
};

class Book {
public:
    string title, ISBN;
    int year;
    Author* author;
    vector<Edition> editions;

    Book(string t, string i, int y, Author* a) : title(t), ISBN(i), year(y),
author(a) {}
    void addEdition(string lang, int year) {
        editions.push_back(Edition(lang, year));
    }
};
```

Lab # 02

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

class Department;

class Employee {
public:
    int id;
    string name;
    vector<Department*> departments;
    Employee(int i, string n) : id(i), name(n) {}
};

class Department {
public:
    string name, location;
    Employee* manager;
    Department(string n, string loc, Employee* m) : name(n), location(loc),
manager(m) {}
};
```

Lab # 03

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

class Cylinder
{
public:
    float diameter;
    string material;
    Cylinder(float d, string m) : diameter(d), material(m) {}
};

class Engine
{
public:
    string engineNumber;
    int horsePower;
    vector<Cylinder> cylinders;
    Engine(string num, int hp) : engineNumber(num), horsePower(hp) {}
};

class Tire
{
public:
    string brand;
    float treadDepth;
    Tire(string b, float td) : brand(b), treadDepth(td) {}
};

class Wheel
{
public:
    int size;
    string type;
    Tire tire;
    Wheel(int s, string t, Tire ti) : size(s), type(t), tire(ti) {}
};

class Seat
{
public:
    string material;
    bool isHeated;
    Seat(string m, bool h) : material(m), isHeated(h) {}
};

class Car
```

```
{
public:
    string model, make;
    int year;
    Engine engine;
    vector<Wheel> wheels;
    vector<Seat> seats;

    Car(string mo, string ma, int y, Engine e) : model(mo), make(ma), year(y),
engine(e) {}
};
```

Lab # 04

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

class Page
{
public:
    int number;
    string content;
    Page(int n, string c) : number(n), content(c) {}
};

class Chapter
{
public:
    int number;
    string title, summary;
    vector<Page> pages;
    Chapter(int n, string t, string s) : number(n), title(t), summary(s) {}
    void addPage(int n, string c)
    {
        pages.emplace_back(n, c);
    }
};

class Book
{
public:
    string title, ISBN, author;
    vector<Chapter> chapters;
    Book(string t, string i, string a) : title(t), ISBN(i), author(a) {}
    void addChapter(Chapter c)
    {
        chapters.push_back(c);
    }
};
```

Lab # 05

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

class Account
{
public:
    int accNumber;
    float balance;
    Account(int num) : accNumber(num), balance(0) {}
    virtual void deposit(float amt) { balance += amt; }
    virtual void withdraw(float amt) { balance -= amt; }
    virtual void statement() const
    {
        cout << "Acc#: " << accNumber << ", Balance: " << balance << "\n";
    }
};

class SavingsAccount : public Account
{
public:
    SavingsAccount(int num) : Account(num) {}
};

class CurrentAccount : public Account
{
public:
    CurrentAccount(int num) : Account(num) {}
};

class Customer
{
public:
    int id;
    string name;
    vector<Account *> accounts;
    Customer(int i, string n) : id(i), name(n) {}
    void addAccount(Account *a) { accounts.push_back(a); }
};
```

Lab # 06

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

class Person
{
public:
    int id;
    string name;
    Person(int i, string n) : id(i), name(n) {}
};

class Course
{
public:
    string code, title;
    Person *teacher;
    Course(string c, string t, Person *teach) : code(c), title(t),
teacher(teach) {}
};

class Transcript
{
public:
    Course *course;
    string grade;
    Transcript(Course *c, string g) : course(c), grade(g) {}
};

class Student : public Person
{
public:
    vector<Course *> enrolled;
    vector<Transcript> transcript;
    Student(int i, string n) : Person(i, n) {}
    void enroll(Course *c) { enrolled.push_back(c); }
    void addGrade(Course *c, string g) { transcript.emplace_back(c, g); }
};

class Teacher : public Person
{
public:
    string dept;
    vector<Course *> teaches;
    Teacher(int i, string n, string d) : Person(i, n), dept(d) {}
};
```

```
class Department
{
public:
    string name;
    vector<Teacher *> teachers;
    vector<Course *> courses;
};

class Appointment;

class Patient
{
public:
    string name, history;
    vector<Appointment *> appointments;
};

class Doctor
{
public:
    string name, department;
};

class Nurse
{
public:
    string name;
    vector<Patient *> patients;
    vector<Doctor *> doctors;
};

class Appointment
{
public:
    Patient *patient;
    Doctor *doctor;
};
```

Task # 01

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

using namespace std;

class Complex {
private:
    float real;
    float imag;

public:
    // Constructor
    Complex(float r = 0, float i = 0) : real(r), imag(i) {}

    // Overload +
    Complex operator+(const Complex& other) const {
        return Complex(real + other.real, imag + other.imag);
    }

    // Overload *
    Complex operator*(const Complex& other) const {
        float r = (real * other.real) - (imag * other.imag);
        float i = (real * other.imag) + (imag * other.real);
        return Complex(r, i);
    }
    // Display
    void display() const {
        cout << real << (imag >= 0 ? " + " : " - ") << abs(imag) << "i" <<
endl;
    }
};

int main() {
    Complex c1(3, 2);     // 3 + 2i
    Complex c2(1, 7);     // 1 + 7i

    Complex sum = c1 + c2;
    Complex product = c1 * c2;

    cout << "c1 = "; c1.display();
    cout << "c2 = "; c2.display();

    cout << "\nSum (c1 + c2) = "; sum.display();
    cout << "Product (c1 * c2) = "; product.display();

    return 0;
}
```

Task # 02

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

// Base class
class Transport {
public:
    virtual float calculateFare(int distance) = 0;
    virtual ~Transport() {}
};

// Derived class: Bus
class Bus : public Transport {
public:
    float calculateFare(int distance) override {
        return distance * 1.5;
    }
};

class Taxi : public Transport {
public:
    float calculateFare(int distance) override {
        return 100 + (distance * 5);
    }
};

int main() {
    int distance;
    cout << "Enter travel distance in km: ";
    cin >> distance;

    Transport* vehicle;

    // Bus Fare
    Bus bus;
    vehicle = &bus;
    cout << "Bus Fare: Rs. " << vehicle->calculateFare(distance) << endl;

    // Taxi Fare
    Taxi taxi;
    vehicle = &taxi;
    cout << "Taxi Fare: Rs. " << vehicle->calculateFare(distance) << endl;

    return 0;
}
```

Task # 03

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

void generateInvoice(string itemName, int quantity, float pricePerUnit) {
    float total = quantity * pricePerUnit;
    cout << fixed << setprecision(2);
    cout << "\nInvoice (Unit-based):" << endl;
    cout << "Item: " << itemName << endl;
    cout << "Quantity: " << quantity << " x $" << pricePerUnit << endl;
    cout << "Total: $" << total << endl;
}

void generateInvoice(string itemName, float weightKg, float pricePerKg) {
    float total = weightKg * pricePerKg;
    cout << fixed << setprecision(2);
    cout << "\nInvoice (Weight-based):" << endl;
    cout << "Item: " << itemName << endl;
    cout << "Weight: " << weightKg << " kg x $" << pricePerKg << endl;
    cout << "Total: $" << total << endl;
}

int main() {
    generateInvoice("Pen", 3, 2.0);
    generateInvoice("Apples", 1.5f, 4.0f);

    return 0;
}
```

Task # 04

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

class Date {
private:
    int day, month, year;

public:
    Date(int d, int m, int y) : day(d), month(m), year(y) {}

    // Overload > operator
    bool operator>(const Date& other) const {
        if (year > other.year)
            return true;
        else if (year == other.year && month > other.month)
            return true;
        else if (year == other.year && month == other.month && day >
other.day)
            return true;
        return false;
    }

    // Overload == operator
    bool operator==(const Date& other) const {
        return (day == other.day && month == other.month && year ==
other.year);
    }

    void display() const {
        cout << day << "/" << month << "/" << year;
    }
};

int main() {
    Date date1(15, 5, 2025);
    Date date2(10, 6, 2025);

    cout << "Date 1: ";
    date1.display();
    cout << "\nDate 2: ";
    date2.display();
    cout << "\n\n";

    if (date1 > date2)
        cout << "Date 1 is later than Date 2\n";
    else if (date1 == date2)
        cout << "Both dates are the same\n";
```

```
    else
        cout << "Date 1 is earlier than Date 2\n";

    return 0;
}
```

Task # 05

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

class Employee {
protected:
    string name;
    float salary;

public:
    Employee(string n, float s) : name(n), salary(s) {}

    virtual void calculateBonus() {
        cout << name << ": Bonus not defined.\n";
    }

    virtual ~Employee() {}
};

// Derived class: Manager
class Manager : public Employee {
private:
    float performanceRating;

public:
    Manager(string n, float s, float rating)
        : Employee(n, s), performanceRating(rating) {}

    void calculateBonus() override {
        float bonus = salary * (performanceRating / 10);
        cout << name << " (Manager) Bonus: $" << bonus << endl;
    }
};

// Derived class: Engineer
class Engineer : public Employee {
private:
    int skillLevel;

public:
    Engineer(string n, float s, int level)
        : Employee(n, s), skillLevel(level) {}

    void calculateBonus() override {
        float bonus = skillLevel * 500;
        cout << name << " (Engineer) Bonus: $" << bonus << endl;
    }
};
```

```
class Intern : public Employee {
public:
    Intern(string n, float s) : Employee(n, s) {}

    void calculateBonus() override {
        cout << name << " (Intern) Bonus: $0 (Not eligible)\n";
    }
};

int main() {
    Employee* emp1 = new Manager("Alice", 80000, 9.2);
    Employee* emp2 = new Engineer("Bob", 60000, 4);
    Employee* emp3 = new Intern("Charlie", 20000);

    emp1->calculateBonus();
    emp2->calculateBonus();
    emp3->calculateBonus();

    // Clean
    delete emp1;
    delete emp2;
    delete emp3;

    return 0;
}
```

Task # 01

```cpp
#include <iostream>
#include <string>

using namespace std;

class Course {
public:
    virtual void registerCourse() = 0;
};

class MathCourse : public Course {
public:
    void registerCourse() {
        cout << "Maths course has been registered!" << endl;
    }
};

class ProgrammingCourse : public Course {
public:
    void registerCourse() {
        cout << "Programming course has been registered!" << endl;
    }
};

int main() {
    Course *c1 = new MathCourse();
    Course *c2 = new ProgrammingCourse();

    c1->registerCourse();
    c2->registerCourse();
    return 0;
}
```

Task # 02

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

// Base class
class SmartDevice {
public:
    virtual void turnOn() = 0;
    virtual void turnOff() = 0;
    virtual ~SmartDevice() {}
};

// Derived class for Smart Light
class SmartLight : public SmartDevice {
public:
    void turnOn() override {
        cout << "Smart Light is turned ON.\n";
    }

    void turnOff() override {
        cout << "Smart Light is turned OFF.\n";
    }
};

// Derived class for Smart Fan
class SmartFan : public SmartDevice {
public:
    void turnOn() override {
        cout << "Smart Fan is spinning now.\n";
    }

    void turnOff() override {
        cout << "Smart Fan is stopped.\n";
    }
};

int main() {
    SmartDevice* device;

    SmartLight light;
    SmartFan fan;

    device = &light;
    device->turnOn();
    device->turnOff();

    cout << endl;
```

```
    device = &fan;
    device->turnOn();
    device->turnOff();

    return 0;
}
```

Task # 03

```cpp
#include <iostream>
#include <stdexcept> // runtime error
using namespace std;

// Function to validate marks
void validateMarks(int marks) {
    if (marks < 0 || marks > 100) {
        throw runtime_error("Invalid input: Marks should be between 0 and
100");
    }
}

int main() {
    int marks;

    cout << "Enter marks for the subject (0 to 100): ";
    cin >> marks;

    try {
        validateMarks(marks);
        cout << "Marks entered: " << marks << endl;
    }
    catch (const runtime_error& e) {
        cout << "Error: " << e.what() << endl;
    }

    return 0;
}
```

Task # 04

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

class FoodOrder {
public:
    void placeOrder() {
        cout << "Placing order...\n";
        checkStock();
        cookFood();
        generateInvoice();
        cout << "Order placed successfully!\n";
    }

private:
    void checkStock() {
        cout << "Checking stock...\n";
    }

    void cookFood() {
        cout << "Cooking food...\n";
    }

    void generateInvoice() {
        cout << "Generating invoice...\n";
    }
};

int main() {
    FoodOrder order;
    order.placeOrder();
    return 0;
}
```

Task # 05

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

void checkEligibility(int age) {
    if (age < 18) {
        throw runtime_error("Age is below 18. Not eligible for license.");
    }
    cout << "Eligible for driving license!" << endl;
}

int main() {
    int age;
    cout << "Enter your age: ";
    cin >> age;

    try {
        checkEligibility(age);
    }
    catch (const runtime_error& e) {
        cout << "Error: " << e.what() << endl;
    }

    return 0;
}
```