



SAVITRIBAI PHULE PUNE UNIVERSITY

**S. Y. B. B. A. (C.A.) SEMESTER IV
(CBCS 2019 PATTERN)**

PRACTICAL SLIP

NAME : LALIT DEVIDAS PATIL

**COLLEGE NAME: SINHGAD COLLEGE OF ARTS &
COMMERCE WARJE PUNE-58**

ROLL NO : 106 DIVISION:B SEAT NO:

--	--	--	--	--

ACADEMIC YEAR : 2023-24

Certificate

**This is to certify that
Mr. PATIL LALIT DEVIDAS**

**Seat Number_____of S.Y.BBA(CA) Sem- IV has
Successfully completed Laboratory course
(C++) in the Year . He has scored mark out of 10
(For Lab Book).**

Subject Teacher

H.O.D./Coordinator

Internal Examiner

External Examiner

Slip 11

A) Create A C++ Class Myarray ,Which Constains Single Dimensional Integer Array Of Given Size. Write A Member Function To Display Sum Of Given Array Elements (Use Dynamic Constuctor And Destructor)...

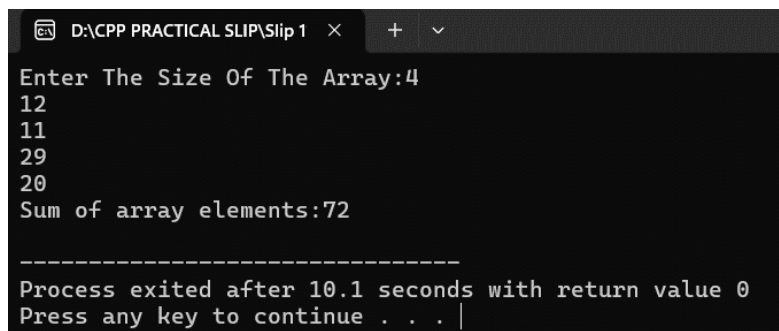
```
#include <iostream>
class MyArray {
    private:
        int *arr;
        int size;
    public:
        MyArray(int s){
            size = s;
            arr= new int[size];
        }
        MyArray(){
            delete[] arr;
        }
        void displaySum(){
            int sum=0;
            for(int i=0; i<size;++i){
                sum += arr[i];
            }
            std::cout<<"Sum of array elements:"<<sum
<<std::endl;
        }
        void setValue(int index, int value){
            if(index >=0 && index < size){
                arr[index]=value;
            }else{
                std::cerr<<"Error: Index out Of bounds."
<<std::endl;
            }
        }
        int getValue(int index){
            if (index >= 0 && index < size){
                return arr[index];
            } else {
                std::cerr << "Error: Index out Of bounds." <<std::endl;
            }
        }
    };
};
```

```

        return -1;
    }
};

int main(){
    int size;
    std::cout<<"Enter The Size Of The Array:";
    std::cin >> size ;
    MyArray myArr(size);
    for(int i = 0; i<size; ++i){
        int value;
        std::cin >> value;
        myArr.setValue(i,value);
    }
    myArr.displaySum();
    return 0;
}

```



```

D:\C++ PRACTICAL SLIP\Slip 1
Enter The Size Of The Array:4
12
11
29
20
Sum of array elements:72

-----
Process exited after 10.1 seconds with return value 0
Press any key to continue . . . |

```

B) Create A Base Class Student With Data Members Roll No, Name. Derives Two Classes From It, Class Theory With Data Members M1, M2, M3, M4 And Class Practical With Data Members P1, P2. Class Result(Total_Marks, Percentage, Grade) Inherits Both Theory And Practical Classes. (Use Concept Virtual Base Class And Protected Access Specifiers)

Write A C++ Menu Driven Program To Perform The Following Functions:

- i. Accept Student Information**
- ii. Display Student Information**
- iii. Calculate Total_Marks, Percentage And Grade.**

```
#include <iostream>
#include <string>
using namespace std;
class Student {
protected:
    int rollNo;
    string name;
public:
    void acceptInfo() {
        cout << "Enter Roll No: ";
        cin >> rollNo;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
    }
    void displayInfo() {
        cout << "Roll No: " << rollNo << endl;
        cout << "Name: " << name << endl;
    }
};
class Theory : virtual public Student {
protected:
    int m1, m2, m3, m4;

public:
    void acceptTheoryMarks() {
```

```

        cout << "Enter Theory Marks (M1 M2 M3 M4): ";
        cin >> m1 >> m2 >> m3 >> m4;
    }
    void displayTheoryMarks() {
        cout << "Theory Marks: " << m1 << " " << m2 << " " << m3
<< " " << m4 << endl;
    }
};
class Practical : virtual public Student {
protected:
    int p1, p2;
public:
    void acceptPracticalMarks() {
        cout << "Enter Practical Marks (P1 P2): ";
        cin >> p1 >> p2;
    }
    void displayPracticalMarks() {
        cout << "Practical Marks: " << p1 << " " << p2 << endl;
    }
};
class Result : public Theory, public Practical {
private:
    int totalMarks;
    float percentage;
    char grade;
public:
    void calculateResult() {
        totalMarks = m1 + m2 + m3 + m4 + p1 + p2;
        percentage = static_cast<float>(totalMarks) / 6;

        if (percentage >= 90)
            grade = 'A';
        else if (percentage >= 80)
            grade = 'B';
        else if (percentage >= 70)
            grade = 'C';
        else if (percentage >= 60)
            grade = 'D';
        else if (percentage >= 50)

```

```

        grade = 'E';
    else
        grade = 'F'; // Fail
    }
    void displayResult() {
        cout << "Total Marks: " << totalMarks << endl;
        cout << "Percentage: " << percentage << "%" << endl;
        cout << "Grade: " << grade << endl;
    }
};
int main() {
    Result studentResult;
    int choice;
    do {
        cout << "\nMenu:";
        cout << "\n1. Accept Student Information";
        cout << "\n2. Display Student Information";
        cout << "\n3. Calculate Result";
        cout << "\n4. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                studentResult.acceptInfo();
                studentResult.acceptTheoryMarks();
                studentResult.acceptPracticalMarks();
                break;
            case 2:
                studentResult.displayInfo();
                studentResult.displayTheoryMarks();
                studentResult.displayPracticalMarks();
                break;
            case 3:
                studentResult.calculateResult();
                studentResult.displayResult();
                break;
            case 4:
                cout << "Exiting...";

```

```

        break;
    default:
        cout << "Invalid choice! Please enter a valid option.";
    }
} while (choice != 4);

return 0;
}

```

```

D:\CPP PRACTICAL SLIP\Slip 1
Menu:
1. Accept Student Information
2. Display Student Information
3. Calculate Result
4. Exit
Enter your choice: 1
Enter Roll No: 106
Enter Name: LALIT D PATIL
Enter Theory Marks (M1 M2 M3 M4): 95
95
94
94
Enter Practical Marks (P1 P2): 95
95

Menu:
1. Accept Student Information
2. Display Student Information
3. Calculate Result
4. Exit
Enter your choice: 2
Roll No: 106
Name: LALIT D PATIL
Theory Marks: 95 95 94 94
Practical Marks: 95 95

Menu:
1. Accept Student Information
2. Display Student Information
3. Calculate Result
4. Exit
Enter your choice: 3
Total Marks: 568
Percentage: 94.6667%
Grade: A

```


Slip 12

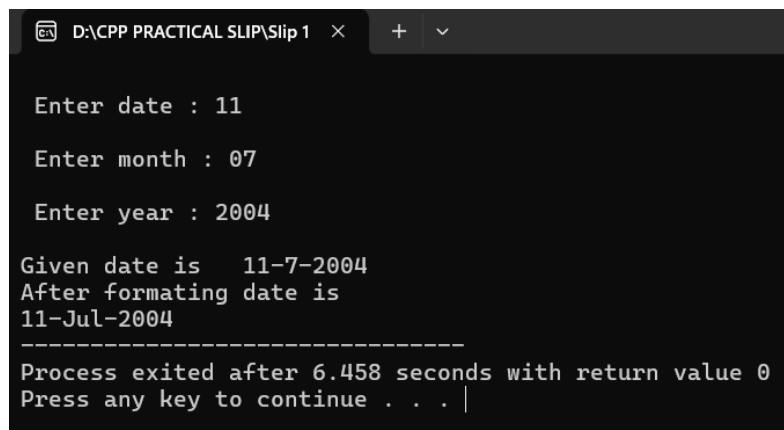
A) Write A C++ Program To Create A Class Date With Data Members Day, Month, And Year. Use Default And Parameterized Constructor To Initialize Date And Display Date In dd-Mon-yyyy Format. (Example: Input:04-01-2021 Output: 04-Jan-2021)

```
#include<iostream>
#include<conio.h>
using namespace std;
class date
{
    int dd,mm,yy;
public:
    date(int d,int m,int y)
    {
        dd=d;
        mm=m;
        yy=y;
    }
    void display()
    {
        cout<<"\nGiven date is\t";
        cout<<dd<<"-"<<mm<<"-"<<yy;
        cout<<"\nAfter formating date is\t";
        switch(mm)
        {
            case 1:
                cout<<"\n"<<dd<<"-Jan-"<<yy; break;
            case 2:
                cout<<"\n"<<dd<<"-Feb-"<<yy; break;
            case 3:
                cout<<"\n"<<dd<<"-Mar-"<<yy; break;
            case 4:
                cout<<"\n"<<dd<<"-Apr-"<<yy; break;
            case 5:
                cout<<"\n"<<dd<<"-May-"<<yy; break;
            case 6:
                cout<<"\n"<<dd<<"-Jun-"<<yy; break;
            case 7:
```

```

        cout<<"\n"<<dd<<"-Jul-"<<yy; break;
    case 8:
        cout<<"\n"<<dd<<"-Aug-"<<yy; break;
    case 9:
        cout<<"\n"<<dd<<"-Sep-"<<yy; break;
    case 10:
        cout<<"\n"<<dd<<"-Oct-"<<yy; break;
    case 11:
        cout<<"\n"<<dd<<"-Nov-"<<yy; break;
    case 12:
        cout<<"\n"<<dd<<"-Dec-"<<yy; break;
    default:
        cout<<"\nInvalid month";
    }
}
}
int main()
{
    int m,dt,y;
    cout<<"\n Enter date : ";
    cin>>dt;
    cout<<"\n Enter month : ";
    cin>>m;
    cout<<"\n Enter year : ";
    cin>>y;
    date d(dt,m,y);
    d.display();
    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 1". The program prompts the user to enter a date, month, and year. The user enters 11 for the date, 07 for the month, and 2004 for the year. The program then displays the formatted date as "11-Jul-2004". At the bottom, it shows the process exit information: "Process exited after 6.458 seconds with return value 0" and "Press any key to continue . . .".

```

D:\CPP PRACTICAL SLIP\Slip 1
Enter date : 11
Enter month : 07
Enter year : 2004
Given date is 11-7-2004
After formatting date is
11-Jul-2004
-----
Process exited after 6.458 seconds with return value 0
Press any key to continue . . .

```

B) Create A C++ Class Weight With Data Members Kilogram, Gram. Write A C++ Program Using Operator Overloading To Perform Following Functions:

- i. To accept weight.**
- ii. To display weight in kilogram and gram format.**
- iii. Overload += operator to add two weights.**
- iv. Overload operator to check equality of two weights.**

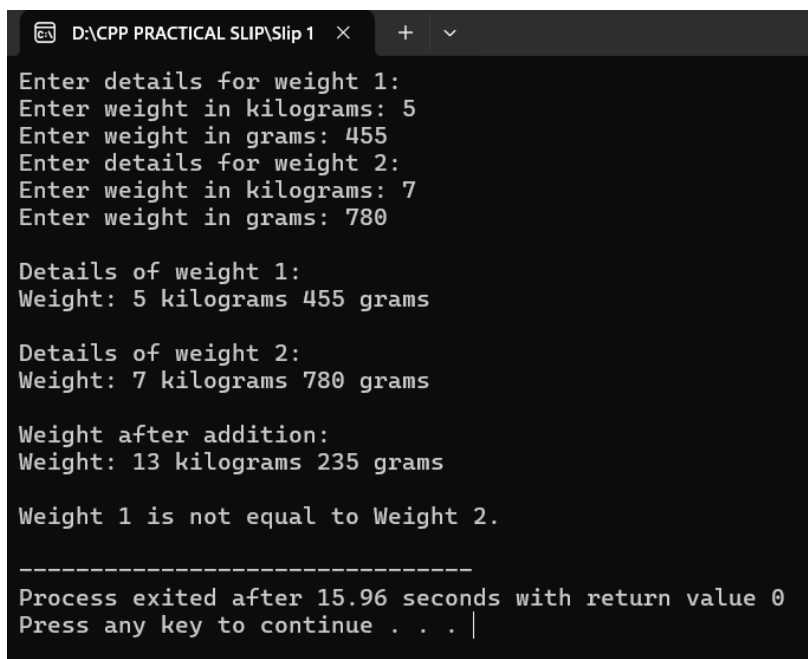
```
#include <iostream>
using namespace std;
class Weight {
private:
    int kilogram;
    int gram;
public:
    Weight(int kg = 0, int g = 0) : kilogram(kg), gram(g) {}
    void acceptWeight() {
        cout << "Enter weight in kilograms: ";
        cin >> kilogram;
        cout << "Enter weight in grams: ";
        cin >> gram;
    }
    void displayWeight() const {
        cout << "Weight: " << kilogram << " kilograms " << gram
        << " grams" << endl;
    }
    Weight& operator+=(const Weight& other) {
        kilogram += other.kilogram;
        gram += other.gram;
        if(gram >= 1000) {
            kilogram += gram / 1000;
            gram %= 1000;
        }
        return *this;
    }
    bool operator==(const Weight& other) const {
        return (kilogram == other.kilogram && gram == other.gram);
    }
};
```

```

int main() {
    Weight w1, w2, w3;
    cout << "Enter details for weight 1:" << endl;
    w1.acceptWeight();
    cout << "Enter details for weight 2:" << endl;
    w2.acceptWeight();
    cout << "\nDetails of weight 1:" << endl;
    w1.displayWeight();
    cout << "\nDetails of weight 2:" << endl;
    w2.displayWeight();
    w3 = w1;
    w3 += w2;
    cout << "\nWeight after addition:" << endl;
    w3.displayWeight();
    if (w1 == w2) {
        cout << "\nWeight 1 is equal to Weight 2." << endl;
    } else {
        cout << "\nWeight 1 is not equal to Weight 2." << endl;
    }

    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 1
Enter details for weight 1:
Enter weight in kilograms: 5
Enter weight in grams: 455
Enter details for weight 2:
Enter weight in kilograms: 7
Enter weight in grams: 780

Details of weight 1:
Weight: 5 kilograms 455 grams

Details of weight 2:
Weight: 7 kilograms 780 grams

Weight after addition:
Weight: 13 kilograms 235 grams

Weight 1 is not equal to Weight 2.

-----
Process exited after 15.96 seconds with return value 0
Press any key to continue . . . |

```

Slip 13

A) Write A C++ Program To Create A Class Product With Data Members Product_Id, Product_Name, Qty, Price. Write Member Functions To Accept And Display Product Information Also Display Number Of Objects Created For Product Class. (Use Static Data Member And Static Member Function)..

```
#include <iostream>
#include <string>
class Product {
private:
    int product_id;
    std::string product_name;
    int qty;
    double price;
    static int object_count;
public:
    Product() {
        product_id = 0;
        product_name = "";
        qty = 0;
        price = 0.0;
        object_count++; }
    Product(int id, std::string name, int quantity, double cost) {
        product_id = id;
        product_name = name;
        qty = quantity;
        price = cost;
        object_count++;}
    void acceptProductInfo() {
        std::cout << "Enter Product ID: ";
        std::cin >> product_id;
        std::cin.ignore(); // to clear the buffer
        std::cout << "Enter Product Name: ";
        std::getline(std::cin, product_name);
        std::cout << "Enter Quantity: ";
        std::cin >> qty;
        std::cout << "Enter Price: ";
        std::cin >> price;
```

```

        object_count++;}
void displayProductInfo() {
    std::cout << "Product ID: " << product_id << std::endl;
    std::cout << "Product Name: " << product_name << std::endl;
    std::cout << "Quantity: " << qty << std::endl;
    std::cout << "Price: " << price << std::endl;}
static int getObjectCount() {
    return object_count;}}};
int Product::object_count = 0;
int main() {
    Product p1, p2, p3;
    std::cout << "Enter details for Product 1:\n";
    p1.acceptProductInfo();
    std::cout << "\nEnter details for Product 2:\n";
    p2.acceptProductInfo();
    std::cout << "\nEnter details for Product 3:\n";
    p3.acceptProductInfo();
    std::cout << "\nProduct Information:\n";
    p1.displayProductInfo();
    p2.displayProductInfo();
    p3.displayProductInfo();
    std::cout << "\nTotal number of products created: " <<
Product::getObjectCount() << std::endl;
return 0; }

```

The screenshot shows a Windows command prompt window titled "D:\C++ PRACTICAL SLIP\Slip 1". The program prompts the user to enter details for three products. The input for Product 1 is ID: 101, Name: Laptop, Quantity: 5, Price: 89000. The input for Product 2 is ID: 102, Name: Smartphone, Quantity: 10, Price: 25000. The input for Product 3 is ID: 103, Name: Earbuds, Quantity: 20, Price: 1500. The program then displays the product information for each product and the total number of products created, which is 6. The program exits after 159 seconds with a return value of 0.

```

D:\C++ PRACTICAL SLIP\Slip 1 x + v
Enter details for Product 1:
Enter Product ID: 101
Enter Product Name: Laptop
Enter Quantity: 5
Enter Price: 89000

Enter details for Product 2:
Enter Product ID: 102
Enter Product Name: Smartphone
Enter Quantity: 10
Enter Price: 25000

Enter details for Product 3:
Enter Product ID: 103
Enter Product Name: Earbuds
Enter Quantity: 20
Enter Price: 1500

Product Information:
Product ID: 101
Product Name: Laptop
Quantity: 5
Price: 89000
Product ID: 102
Product Name: Smartphone
Quantity: 10
Price: 25000
Product ID: 103
Product Name: Earbuds
Quantity: 20
Price: 1500

Total number of products created: 6

-----
Process exited after 159 seconds with return value 0
Press any key to continue . . . |

```

B) Create a C++ class Cuboid with data members length, breadth, and height. Write necessary member functions for the following:

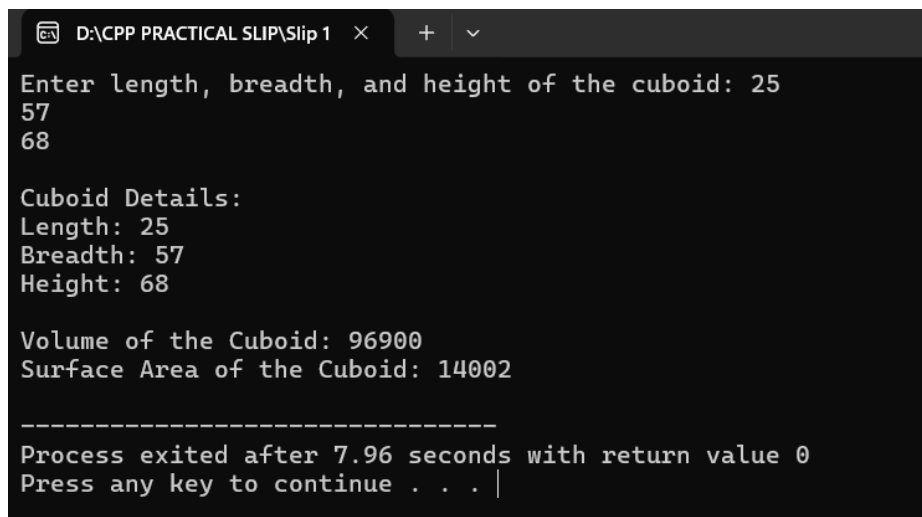
- i. void setvalues(float, float, float) to set values of data members.**
- ii. void getvalues() to display values of data members.**
- iii. float volume() to calculate and return the volume of cuboid.**
- iv. float surface_area() to calculate and return the surface area of cuboid.... (Use Inline function)**

```
#include <iostream>
class Cuboid {
private:
    float length;
    float breadth;
    float height;
public:
    void setValues(float l, float b, float h) {
        length = l;
        breadth = b;
        height = h;
    }
    void getValues() {
        std::cout << "Length: " << length << std::endl;
        std::cout << "Breadth: " << breadth << std::endl;
        std::cout << "Height: " << height << std::endl;
    }
    float volume() {
        return length * breadth * height;
    }
    inline float surfaceArea() {
        return 2 * (length * breadth + length * height + breadth *
height);
    }
};
int main() {
    Cuboid cuboid;
    float l, b, h;
    std::cout << "Enter length, breadth, and height of the cuboid: ";
    std::cin >> l >> b >> h;
```

```

    cuboid.setValues(l, b, h);
    std::cout << "\nCuboid Details:\n";
    cuboid.getValues();
    std::cout << "\nVolume of the Cuboid: " << cuboid.volume() <<
std::endl;
    std::cout << "Surface Area of the Cuboid: " <<
cuboid.surfaceArea() << std::endl;
    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 1". The program prompts the user to "Enter length, breadth, and height of the cuboid:" and receives the inputs 25, 57, and 68. It then displays the "Cuboid Details" with Length: 25, Breadth: 57, and Height: 68. Below this, it calculates and displays "Volume of the Cuboid: 96900" and "Surface Area of the Cuboid: 14002". The program ends with a separator line, a message "Process exited after 7.96 seconds with return value 0", and a prompt "Press any key to continue . . . |".

```

D:\CPP PRACTICAL SLIP\Slip 1
Enter length, breadth, and height of the cuboid: 25
57
68

Cuboid Details:
Length: 25
Breadth: 57
Height: 68

Volume of the Cuboid: 96900
Surface Area of the Cuboid: 14002

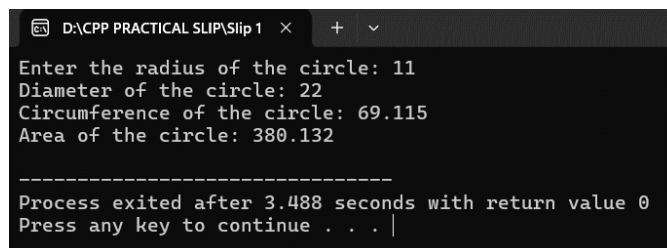
-----
Process exited after 7.96 seconds with return value 0
Press any key to continue . . . |

```


Slip 14

A) Write A C++ Program To Accept Radius Of A Circle. Calculate And Display Diameter, Circumference As Well As Area Of A Circle. (Use Inline Function)

```
#include <iostream>
const float PI = 3.14159;
class Circle {
public:
    inline float diameter(float radius) {
        return 2 * radius;
    }
    inline float circumference(float radius) {
        return 2 * PI * radius;
    }
    inline float area(float radius) {
        return PI * radius * radius;
    }
};
int main() {
    Circle circle;
    float radius;
    std::cout << "Enter the radius of the circle: ";
    std::cin >> radius;
    std::cout << "Diameter of the circle: " << circle.diameter(radius)
<< std::endl;
    std::cout << "Circumference of the circle: " <<
circle.circumference(radius) << std::endl;
    std::cout << "Area of the circle: " << circle.area(radius) <<
std::endl;
    return 0;
}
```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 1". The program prompts the user to enter the radius of the circle. The user enters 11. The program then outputs the diameter (22), circumference (69.115), and area (380.132). A separator line is shown, followed by a message indicating the process exited after 3.488 seconds with a return value of 0, and a prompt to press any key to continue.

```
D:\CPP PRACTICAL SLIP\Slip 1 >
Enter the radius of the circle: 11
Diameter of the circle: 22
Circumference of the circle: 69.115
Area of the circle: 380.132

-----
Process exited after 3.488 seconds with return value 0
Press any key to continue . . .
```

B) Create a C++ class MyString with data members a character pointer and str_length. (Use new and delete operator). Write a C++ program using operator overloading to perform following operation:

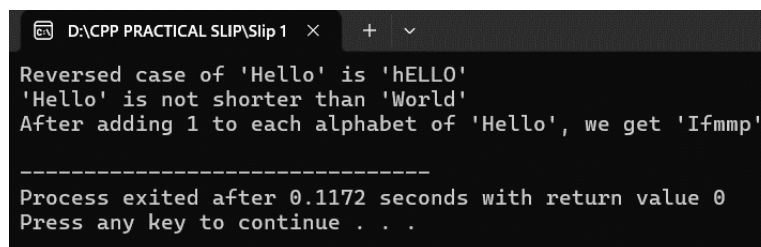
- i. ! To reverse the case of each alphabet from a given string.**
- ii. < To compare length of two strings.**
- iii. + To add constant 'n' to each alphabet of a string.**

```
#include <iostream>
#include <cstring>
class MyString {
private:
    char *str;
    int str_length;
public:
    MyString(const char *s) {
        str_length = strlen(s);
        str = new char[str_length + 1];
        strcpy(str, s); }
    ~MyString() {
        delete[] str; }
    MyString(const MyString &other) {
        str_length = other.str_length;
        str = new char[str_length + 1];
        strcpy(str, other.str); }
    MyString& operator=(const MyString &other) {
        if (this != &other) {
            delete[] str;
            str_length = other.str_length;
            str = new char[str_length + 1];
            strcpy(str, other.str);}
        return *this; }
    MyString operator!() const {
        MyString temp(*this);
        for (int i = 0; i < str_length; ++i) {
            if (isalpha(temp.str[i])) {
                if (islower(temp.str[i])) {
                    temp.str[i] = toupper(temp.str[i]);
                } else {
```

```

        temp.str[i] = tolower(temp.str[i]); }}}
    return temp;}
bool operator<(const MyString &other) const {
    return str_length < other.str_length; }
MyString operator+(int n) const {
    MyString temp(*this);
    for (int i = 0; i < str_length; ++i) {
        if (isalpha(temp.str[i])) {
            temp.str[i] += n;    }}
    return temp;    }
int length() const {
    return str_length; }
const char* c_str() const {
    return str;    }
int main() {
    MyString str1("Hello");
    MyString str2("World");
    MyString reversed = !str1;
    std::cout << "Reversed case of '" << str1.c_str() << "' is '" <<
reversed.c_str() << "'" << std::endl;
    if (str1 < str2) {
        std::cout << "'" << str1.c_str() << "' is shorter than '" <<
str2.c_str() << "'" << std::endl;
    } else {
        std::cout << "'" << str1.c_str() << "' is not shorter than '" <<
str2.c_str() << "'" << std::endl;
    }
    MyString added = str1 + 1;
    std::cout << "After adding 1 to each alphabet of '" << str1.c_str()
<< "', we get '" << added.c_str() << "'" << std::endl;
    return 0;
}

```



The screenshot shows a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 1" with a dark background and light-colored text. The output of the program is as follows:

```

Reversed case of 'Hello' is 'hELLO'
'Hello' is not shorter than 'World'
After adding 1 to each alphabet of 'Hello', we get 'Ifmmp'

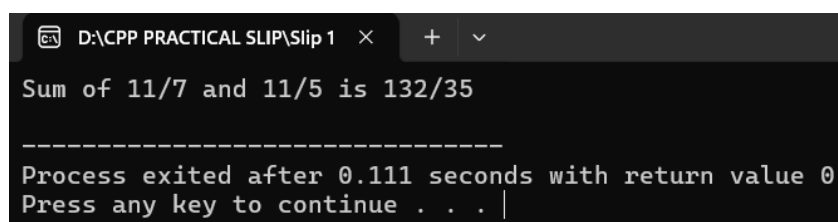
-----
Process exited after 0.1172 seconds with return value 0
Press any key to continue . . .

```

Slip 15

A) Create A C++ Class Fraction With Data Members Numerator And Denominator. Write A C++ Program To Calculate And Display Sum Of Two Fractions. (Use Constructor)...

```
#include <iostream>
class Fraction {
private:
    int numerator;
    int denominator;
public:
    Fraction(int num, int den) : numerator(num), denominator(den) {}
    Fraction add(const Fraction &other) const {
        int common_denominator = denominator * other.denominator;
        int sum_numerator = numerator * other.denominator +
other.numerator * denominator;
        return Fraction(sum_numerator, common_denominator);
    }
    void display() const {
        std::cout << numerator << "/" << denominator;
    }
};
int main() {
    Fraction frac1(11, 07);
    Fraction frac2(11, 05);
    Fraction sum = frac1.add(frac2);
    std::cout << "Sum of ";
    frac1.display();
    std::cout << " and ";
    frac2.display();
    std::cout << " is ";
    sum.display();
    std::cout << std::endl;
    return 0;}
```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 1". The output of the program is displayed as follows:

```
Sum of 11/7 and 11/5 is 132/35

-----
Process exited after 0.111 seconds with return value 0
Press any key to continue . . . |
```

B) Write A C++ Class Seller (S_Name, Product_Name, Sales_Quantity, Target_Quantity, Month, Commission). Each Salesman Deals With A Separate Product And Is Assigned A Target For A Month. At The End Of The Month His Monthly Sales Is Compared With Target And Commission Is Calculated As Follow

- **If Sales_Quantity > Target_Quantity Then Commission Is 25% Of Extra Sales Made + 10% Of Target.**
- **If Sales_Quantity = Target_Quantity Then Commission Is 10% Of Target.**
- **Otherwise Commission Is Zero.**

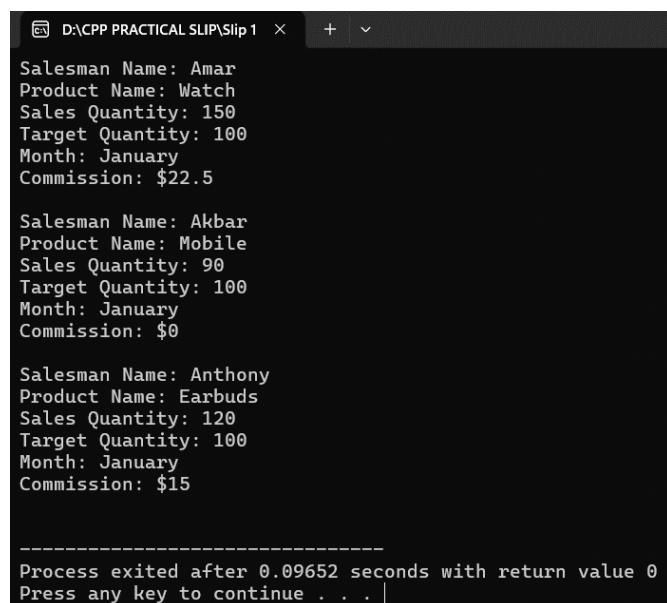
Display Salesman Information Along With Commission Obtained. (Use Array Of Objects)

```
#include <iostream>
#include <string>
class Seller {
private:
    std::string S_Name;
    std::string Product_Name;
    int Sales_Quantity;
    int Target_Quantity;
    std::string Month;
    double Commission;
public:
    Seller(std::string name, std::string product, int salesQty, int
targetQty, std::string month)
        : S_Name(name), Product_Name(product),
Sales_Quantity(salesQty), Target_Quantity(targetQty),
Month(month), Commission(0.0) {}
    void calculateCommission() {
        if (Sales_Quantity > Target_Quantity) {
            Commission = 0.25 * (Sales_Quantity - Target_Quantity) +
0.10 * Target_Quantity;
        } else if (Sales_Quantity == Target_Quantity) {
            Commission = 0.10 * Target_Quantity;
        } else {
            Commission = 0.0;
        }
    }
};
```

```

    }
}
void display() {
    std::cout << "Salesman Name: " << S_Name << std::endl;
    std::cout << "Product Name: " << Product_Name << std::endl;
    std::cout << "Sales Quantity: " << Sales_Quantity << std::endl;
    std::cout << "Target Quantity: " << Target_Quantity <<
std::endl;
    std::cout << "Month: " << Month << std::endl;
    std::cout << "Commission: $" << Commission << std::endl;
    std::cout << std::endl;
}
};
int main() {
    const int numSellers = 3;
    Seller sellers[numSellers] = {
        Seller("Amar", "Watch", 150, 100, "January"),
        Seller("Akbar", "Mobile", 90, 100, "January"),
        Seller("Anthony", "Earbuds", 120, 100, "January")
    };
    for (int i = 0; i < numSellers; ++i) {
        sellers[i].calculateCommission();
    }
    for (int i = 0; i < numSellers; ++i) {
        sellers[i].display();
    }
    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 1 x + v
Salesman Name: Amar
Product Name: Watch
Sales Quantity: 150
Target Quantity: 100
Month: January
Commission: $22.5

Salesman Name: Akbar
Product Name: Mobile
Sales Quantity: 90
Target Quantity: 100
Month: January
Commission: $0

Salesman Name: Anthony
Product Name: Earbuds
Sales Quantity: 120
Target Quantity: 100
Month: January
Commission: $15

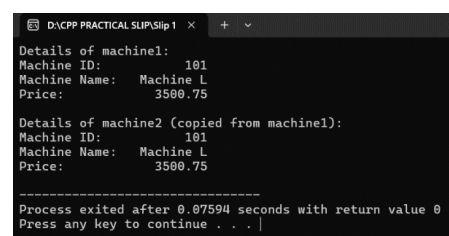
-----
Process exited after 0.09652 seconds with return value 0
Press any key to continue . . . |

```

Slip 16

A) Write A C++ Program To Create A Class Machine With Data Members Machine_Id, Machine_Name, Price. Create And Initialize All Values Of Machine Object By Using Parameterized Constructor And Copy Constructor. Display Details Of Machine Using Setw() And Setprecision().

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class Machine {
private:
    int Machine_Id;
    string Machine_Name;
    double Price;
public:
    Machine(int id, const string& name, double price) :
Machine_Id(id), Machine_Name(name), Price(price) {}
    Machine(const Machine& other) : Machine_Id(other.Machine_Id),
Machine_Name(other.Machine_Name), Price(other.Price) {}
    void displayDetails() {
        cout << setw(15) << left << "Machine ID:" << setw(10) <<
right << Machine_Id << endl;
        cout << setw(15) << left << "Machine Name:" << setw(10) <<
right << Machine_Name << endl;
        cout << setw(15) << left << "Price:" << setw(10) << right <<
fixed << setprecision(2) << Price << endl;
    };
int main() {
    Machine machine1(101, "Machine L", 3500.75);
    Machine machine2 = machine1;
    cout << "Details of machine1:" << endl;
    machine1.displayDetails();
    cout << "\nDetails of machine2 (copied from machine1):" <<
endl;
    machine2.displayDetails();
    return 0;}
```



```
D:\C++ PRACTICAL SLIP\Slip1 x + v
Details of machine1:
Machine ID:      101
Machine Name:    Machine L
Price:           3500.75

Details of machine2 (copied from machine1):
Machine ID:      101
Machine Name:    Machine L
Price:           3500.75

-----
Process exited after 0.07594 seconds with return value 0
Press any key to continue . . . |
```

B) Create a C++ class MyMatrix and Write necessary member functions for the following:

- i. To accept a Matrix**
- ii. To display a Matrix**
- iii. Overload unary - operator to calculate transpose of a Matrix.**
- iv. Overload unary '++' operator to increment matrix elements by 1.**

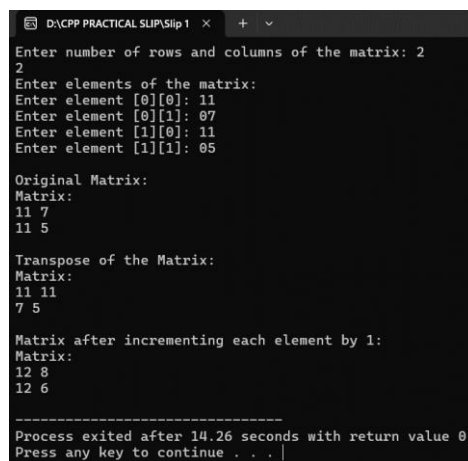
```
#include <iostream>
#include <vector>
class MyMatrix {
private:
    std::vector<std::vector<int> > matrix;
    int rows;
    int cols;
public:
    MyMatrix(int rows, int cols) : rows(rows), cols(cols) {
        matrix.resize(rows, std::vector<int>(cols, 0));
    }
    void acceptMatrix() {
        std::cout << "Enter elements of the matrix:\n";
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                std::cout << "Enter element [" << i << "][" << j << "]:
";
                std::cin >> matrix[i][j];
            } } }
    void displayMatrix() {
        std::cout << "Matrix:\n";
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                std::cout << matrix[i][j] << " "; }
            std::cout << "\n";
        } }
    MyMatrix operator-() {
        MyMatrix transpose(cols, rows);
        for (int i = 0; i < cols; ++i) {
            for (int j = 0; j < rows; ++j) {
```



```

        transpose.matrix[i][j] = matrix[j][i];
    } }
    return transpose;
}
MyMatrix operator++() {
    MyMatrix result(rows, cols);
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result.matrix[i][j] = matrix[i][j] + 1;
        }
    }
    return result;
}
};
int main() {
    int rows, cols;
    std::cout << "Enter number of rows and columns of the matrix: ";
    std::cin >> rows >> cols;
    MyMatrix mat(rows, cols);
    mat.acceptMatrix();
    std::cout << "\nOriginal Matrix:\n";
    mat.displayMatrix();
    MyMatrix transposedMat = -mat;
    std::cout << "\nTranspose of the Matrix:\n";
    transposedMat.displayMatrix();
    MyMatrix incrementedMat = ++mat;
    std::cout << "\nMatrix after incrementing each element by 1:\n";
    incrementedMat.displayMatrix();
    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 1". The program prompts the user to enter the number of rows and columns, which are both 2. It then prompts for the elements of the matrix, which are entered as 11, 7, 11, and 5. The program displays the original matrix as a 2x2 grid. It then calculates the transpose of the matrix and displays it. Finally, it increments each element of the original matrix by 1 and displays the resulting matrix. The program exits after 14.26 seconds with a return value of 0.

```

D:\CPP PRACTICAL SLIP\Slip 1 x + v
Enter number of rows and columns of the matrix: 2
2
Enter elements of the matrix:
Enter element [0][0]: 11
Enter element [0][1]: 07
Enter element [1][0]: 11
Enter element [1][1]: 05

Original Matrix:
Matrix:
11 7
11 5

Transpose of the Matrix:
Matrix:
11 11
7 5

Matrix after incrementing each element by 1:
Matrix:
12 8
12 6

-----
Process exited after 14.26 seconds with return value 0
Press any key to continue . . . |

```

Slip 17

A) Create A C++ Class Mymatrix. Write A C++ Program To Accept And Display A Matrix. Overload Binary '-' Operator To Calculate Subtraction Of Two Matrices.

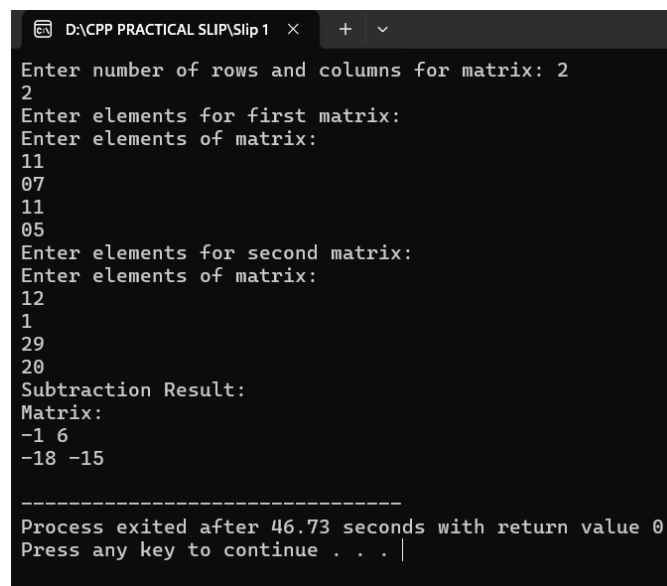
```
#include <iostream>
#include <vector>
class Mymatrix {
private:
    std::vector<std::vector<int> > matrix;
    int rows;
    int cols;
public:
    Mymatrix(int r, int c) : rows(r), cols(c) {
        matrix.resize(rows);
        for (int i = 0; i < rows; ++i) {
            matrix[i].resize(cols);
        }
    }
    void inputMatrix() {
        std::cout << "Enter elements of matrix:" << std::endl;
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                std::cin >> matrix[i][j];
            }
        }
    }
    void displayMatrix() {
        std::cout << "Matrix:" << std::endl;
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                std::cout << matrix[i][j] << " ";
            }
            std::cout << std::endl;
        }
    }
    Mymatrix operator-(const Mymatrix &other) {
        Mymatrix result(rows, cols);
        for (int i = 0; i < rows; ++i) {
```

```

        for (int j = 0; j < cols; ++j) {
            result.matrix[i][j] = matrix[i][j] - other.matrix[i][j];
        }
    }
    return result;
}
};

int main() {
    int rows, cols;
    std::cout << "Enter number of rows and columns for matrix: ";
    std::cin >> rows >> cols;
    Mymatrix matrix1(rows, cols);
    std::cout << "Enter elements for first matrix:" << std::endl;
    matrix1.inputMatrix();
    Mymatrix matrix2(rows, cols);
    std::cout << "Enter elements for second matrix:" << std::endl;
    matrix2.inputMatrix();
    Mymatrix result = matrix1 - matrix2;
    std::cout << "Subtraction Result:" << std::endl;
    result.displayMatrix();
    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\C++ PRACTICAL SLIP\Slip 1". The program prompts the user to enter the number of rows and columns for a matrix, which is set to 2. It then prompts for the elements of the first matrix, which are 11, 07, 11, and 05. Next, it prompts for the elements of the second matrix, which are 12, 1, 29, and 20. The program then displays the subtraction result matrix, which contains the values -1, 6, -18, and -15. At the bottom, it shows the process exit time and a prompt to press any key to continue.

```

D:\C++ PRACTICAL SLIP\Slip 1 x + v
Enter number of rows and columns for matrix: 2
2
Enter elements for first matrix:
Enter elements of matrix:
11
07
11
05
Enter elements for second matrix:
Enter elements of matrix:
12
1
29
20
Subtraction Result:
Matrix:
-1 6
-18 -15

-----
Process exited after 46.73 seconds with return value 0
Press any key to continue . . . |

```

B) Design Two Base Classes Student (S_Id, Name, Class) And Competition (C_Id, C_Name). Derive A Class Stud_Comp(Rank) From It. Write A Menu Driven Program To Perform Following Functions:

- i. Accept Information.**
 - ii. Display Information.**
 - iii. Display Student Details In The Ascending Order Of Rank Of A Specified Competition.**
- (Use Array Of Objects)**

```
#include <iostream>
#include <string>
#include <algorithm>
class Student {
protected:
    int S_id;
    std::string Name;
    std::string Class;
public:
    void acceptStudentInfo() {
        std::cout << "Enter Student ID: ";
        std::cin >> S_id;
        std::cin.ignore();
        std::cout << "Enter Student Name: ";
        std::getline(std::cin, Name);
        std::cout << "Enter Student Class: ";
        std::getline(std::cin, Class);
    }
    void displayStudentInfo() const {
        std::cout << "Student ID: " << S_id << std::endl;
        std::cout << "Student Name: " << Name << std::endl;
        std::cout << "Student Class: " << Class << std::endl;
    }
};
class Competition {
protected:
    int C_id;
    std::string C_Name;
public:
```

```

void acceptCompetitionInfo() {
    std::cout << "Enter Competition ID: ";
    std::cin >> C_id;
    std::cin.ignore();
    std::cout << "Enter Competition Name: ";
    std::getline(std::cin, C_Name);
}
void displayCompetitionInfo() const {
    std::cout << "Competition ID: " << C_id << std::endl;
    std::cout << "Competition Name: " << C_Name << std::endl;
}
int getCompetitionId() const {
    return C_id;
}
};
class StudComp : public Student, public Competition {
private:
    int Rank;
public:
    void acceptStudCompDetails() {
        acceptStudentInfo();
        acceptCompetitionInfo();
        std::cout << "Enter Rank: ";
        std::cin >> Rank;
    }
    void displayStudCompDetails() const {
        displayStudentInfo();
        displayCompetitionInfo();
        std::cout << "Rank: " << Rank << std::endl;
    }
    static bool compareByRank(const StudComp &a, const StudComp
&b) {
        return a.Rank < b.Rank;
    }
};
int main() {
    const int maxSize = 100;
    StudComp students[maxSize];
    int size = 0;

```

```

int choice;
do {
    std::cout << "\nMenu:\n";
    std::cout << "1. Accept Information\n";
    std::cout << "2. Display Information\n";
    std::cout << "3. Display Student Details in ascending order of
Rank of a specified competition\n";
    std::cout << "4. Exit\n";
    std::cout << "Enter your choice: ";
    std::cin >> choice;
    switch (choice) {
    case 1:
        if (size < maxSize) {
            students[size++].acceptStudCompDetails();
        } else {
            std::cout << "Array is full!" << std::endl;
        }
        break;
    case 2:
        std::cout << "\nStudent Details:\n";
        for (int i = 0; i < size; ++i) {
            std::cout << "Student " << i + 1 << ":\n";
            students[i].displayStudCompDetails();
            std::cout << std::endl;
        }
        break;
    case 3:
        int compId;
        std::cout << "Enter the Competition ID: ";
        std::cin >> compId;
        {
            int count = 0;
            StudComp temp[maxSize];
            for (int i = 0; i < size; ++i) {
                if (students[i].getCompetitionId() == compId) {
                    temp[count++] = students[i];
                }
            }
            if (count > 0) {

```

```

        std::sort(temp, temp + count,
StudComp::compareByRank);
        std::cout << "\nStudent Details in ascending order of
Rank for Competition ID " << compId <<
        ":\n";
        for (int i = 0; i < count; ++i) {
            std::cout << "Rank " << i + 1 << ":\n";
            temp[i].displayStudCompDetails();
            std::cout << std::endl;
        }
    } else {
        std::cout << "No students found for Competition ID "
<< compId << std::endl;
    }
}
break;
case 4:
    std::cout << "Exiting...\n";
    break;
default:
    std::cout << "Invalid choice!\n";
    break;
}
} while (choice != 4);
return 0;
}

```

```

D:\C++ PRACTICAL\SLIP\Slip1 x + v
Menu:
1. Accept Information
2. Display Information
3. Display Student Details in ascending order of Rank of a specified competition
4. Exit
Enter your choice: 1
Enter Student ID: 106
Enter Student Name: LALIT D PATIL
Enter Student Class: SYBCA
Enter Competition ID: 1
Enter Competition Name: Poetry
Enter Rank: 1

Menu:
1. Accept Information
2. Display Information
3. Display Student Details in ascending order of Rank of a specified competition
4. Exit
Enter your choice: 2

Student Details:
Student 1:
Student ID: 106
Student Name: LALIT D PATIL
Student Class: SYBCA
Competition ID: 1
Competition Name: Poetry
Rank: 1

Menu:
1. Accept Information
2. Display Information
3. Display Student Details in ascending order of Rank of a specified competition
4. Exit
Enter your choice: |

```

Slip 18

A) Create A C++ Class Student With Data Members Roll_No, S Name, Class, Percentage. Accept Two Students Information And Display Information Of Student Having Maximum Percentage. (Use This Pointer)..

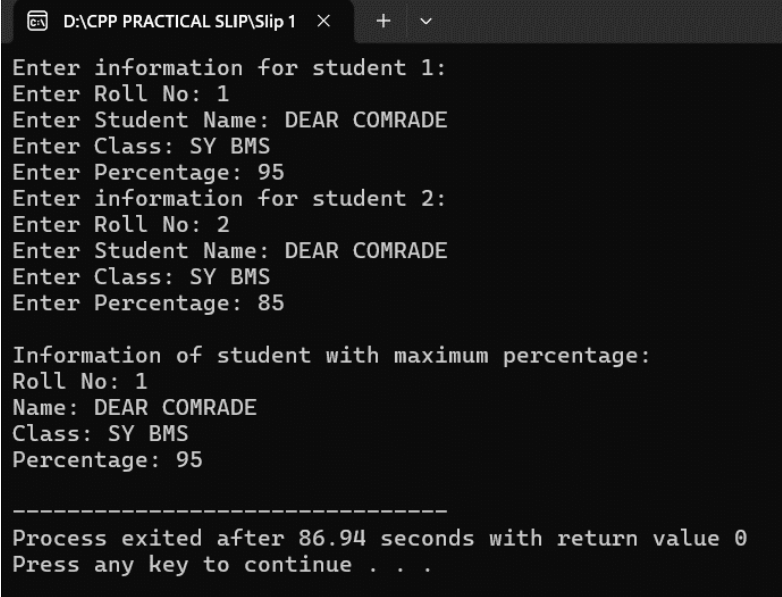
```
#include <iostream>
#include <string>
using namespace std;
class Student {
private:
    int Roll_No;
    string S_Name;
    string Class;
    float Percentage;
public:
    void acceptInfo() {
        cout << "Enter Roll No: ";
        cin >> Roll_No;
        cout << "Enter Student Name: ";
        cin.ignore();
        getline(cin, S_Name);
        cout << "Enter Class: ";
        getline(cin, Class);
        cout << "Enter Percentage: ";
        cin >> Percentage;
    }
    void displayInfo() {
        cout << "Roll No: " << Roll_No << endl;
        cout << "Name: " << S_Name << endl;
        cout << "Class: " << Class << endl;
        cout << "Percentage: " << Percentage << endl;
    }
    Student* comparePercentage(Student* other) {
        if (this->Percentage > other->Percentage)
            return this;
        else
            return other;
    }
};
```



```

    }
};
int main() {
    Student s1, s2;
    cout << "Enter information for student 1:" << endl;
    s1.acceptInfo();
    cout << "Enter information for student 2:" << endl;
    s2.acceptInfo();
    cout << "\nInformation of student with maximum percentage:"
<< endl;
    Student* max_student = s1.comparePercentage(&s2);
    max_student->displayInfo();
    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 1 x + v
Enter information for student 1:
Enter Roll No: 1
Enter Student Name: DEAR COMRADE
Enter Class: SY BMS
Enter Percentage: 95
Enter information for student 2:
Enter Roll No: 2
Enter Student Name: DEAR COMRADE
Enter Class: SY BMS
Enter Percentage: 85

Information of student with maximum percentage:
Roll No: 1
Name: DEAR COMRADE
Class: SY BMS
Percentage: 95

-----
Process exited after 86.94 seconds with return value 0
Press any key to continue . . .

```

B) Create A C++ Class Myarray With Data Members

`int *arr`

`int size`

Write necessary member functions to accept and display Array elements. Overload the following operators:

Operator	Example	Purpose
(Unary)	-A1	Reverse array elements.
+(Binary)	A2-A1+n	Add Constant n to all array elements.

```
#include <iostream>
class MyArray {
private:
    int *arr;
    int size;
public:
    MyArray(int s) : size(s) {
        arr = new int[size]; }
    ~MyArray() {
        delete[] arr; }
    void acceptElements() {
        std::cout << "Enter " << size << " elements:\n";
        for (int i = 0; i < size; ++i) {
            std::cin >> arr[i];
        } }
    void displayElements() {
        std::cout << "Array elements:\n";
        for (int i = 0; i < size; ++i) {
            std::cout << arr[i] << " "; }
        std::cout << std::endl;}
    void operator-() {
        for (int i = 0; i < size / 2; ++i) {
            int temp = arr[i];
            arr[i] = arr[size - i - 1];
            arr[size - i - 1] = temp;
        } }
    MyArray operator+(int n) {
        MyArray result(size);
```

```

        for (int i = 0; i < size; ++i) {
            result.arr[i] = arr[i] + n;    }
        return result; }
int main() {
    int size;
    std::cout << "Enter size of array: ";
    std::cin >> size;
    MyArray array(size);
    array.acceptElements();
    std::cout << "Original array:\n";
    array.displayElements();
    -array;
    std::cout << "Reversed array:\n";
    array.displayElements();
    int constant;
    std::cout << "Enter a constant to add to all elements: ";
    std::cin >> constant;
    MyArray newArray = array + constant;
    std::cout << "Array after adding " << constant << " to each
element:\n";
    newArray.displayElements();
    return 0;
}

```

```

D:\C++ PRACTICAL SLIP\Slip 1 x + -
Enter size of array: 4
Enter 4 elements:
12
11
29
20
Original array:
Array elements:
12 11 29 20
Reversed array:
Array elements:
20 29 11 12
Enter a constant to add to all elements: 11
Array after adding 11 to each element:
Array elements:
31 40 22 23

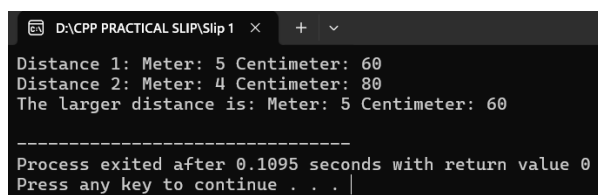
-----
Process exited after 20.72 seconds with return value 0
Press any key to continue . . . |

```

Slip 19

A) Write A C++ Program To Create A Class Distance With Data Members Meter And Centimeter To Represent Distance. Write A Function Larger() To Return The Larger Of Two Distances. (Use This Pointer)

```
#include <iostream>
class Distance {
private:
    int meter;
    int centimeter;
public:
    Distance(int m = 0, int cm = 0) : meter(m), centimeter(cm) {}
    Distance larger(Distance *other) {
        if (this->meter > other->meter || (this->meter == other-
>meter && this->centimeter > other->centimeter))
            return *this;
        else
            return *other;
    }
    void display() {
        std::cout << "Meter: " << meter << " Centimeter: " <<
centimeter << std::endl;
    }
};
int main() {
    Distance d1(5, 60);
    Distance d2(4, 80);
    std::cout << "Distance 1: ";
    d1.display();
    std::cout << "Distance 2: ";
    d2.display();
    Distance largerDistance = d1.larger(&d2);
    std::cout << "The larger distance is: ";
    largerDistance.display();
    return 0;}
```



The screenshot shows a terminal window with the following output:

```
D:\CPP PRACTICAL SLIP\Slip 1 x + v
Distance 1: Meter: 5 Centimeter: 60
Distance 2: Meter: 4 Centimeter: 80
The larger distance is: Meter: 5 Centimeter: 60
-----
Process exited after 0.1095 seconds with return value 0
Press any key to continue . . . |
```

B) Create A C++ Base Class Media. Derive Two Different Classes From It, Class Newspaper With Data Members N_Name, N_Editor, N_Price, No_Of Pages And Class Magazine With Data Members M_Name, M_Editor, M_Price. Write A C++ Program To Accept And Display Information Of Both Newspaper And Magazine. (Use Pure Virtual Function)...

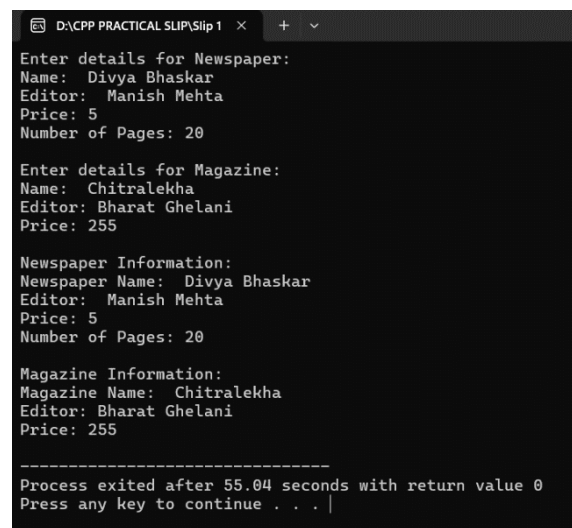
```
#include <iostream>
#include <string>
using namespace std;
class Media {
protected:
    string name;
    string editor;
    float price;
public:
    Media(string n, string e, float p) : name(n), editor(e), price(p) {}
    virtual void displayInfo() = 0; // Pure virtual function};
class Newspaper : public Media {
private:
    string n_name;
    int no_of_pages;
public:
    Newspaper(string n, string e, float p, string nn, int pages)
        : Media(n, e, p), n_name(nn), no_of_pages(pages) {}
    void displayInfo() override {
        cout << "Newspaper Name: " << n_name << endl;
        cout << "Editor: " << editor << endl;
        cout << "Price: " << price << endl;
        cout << "Number of Pages: " << no_of_pages << endl;
    };
};
class Magazine : public Media {
private:
    string m_name;
public:
    Magazine(string n, string e, float p, string mn)
        : Media(n, e, p), m_name(mn) {}
    void displayInfo() override {
        cout << "Magazine Name: " << m_name << endl;
```

```

        cout << "Editor: " << editor << endl;
        cout << "Price: " << price << endl;
    }
}

int main() {
    string nn, ne, mn, me;
    float np, mp;
    int npages;
    cout << "Enter details for Newspaper:\n";
    cout << "Name: ";
    getline(cin, nn);
    cout << "Editor: ";
    getline(cin, ne);
    cout << "Price: ";
    cin >> np;
    cout << "Number of Pages: ";
    cin >> npages;
    cin.ignore();
    cout << "\nEnter details for Magazine:\n";
    cout << "Name: ";
    getline(cin, mn);
    cout << "Editor: ";
    getline(cin, me);
    cout << "Price: ";
    cin >> mp;
    cin.ignore();
    Newspaper newspaper(nn, ne, np, mn, npages);
    Magazine magazine(mn, me, mp, mn);
    cout << "\nNewspaper Information:\n";
    newspaper.displayInfo();
    cout << "\nMagazine Information:\n";
    magazine.displayInfo();
    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 1 x + v
Enter details for Newspaper:
Name: Divya Bhaskar
Editor: Manish Mehta
Price: 5
Number of Pages: 20

Enter details for Magazine:
Name: Chitralekha
Editor: Bharat Ghelani
Price: 255

Newspaper Information:
Newspaper Name: Divya Bhaskar
Editor: Manish Mehta
Price: 5
Number of Pages: 20

Magazine Information:
Magazine Name: Chitralekha
Editor: Bharat Ghelani
Price: 255

-----
Process exited after 55.04 seconds with return value 0
Press any key to continue . . . |

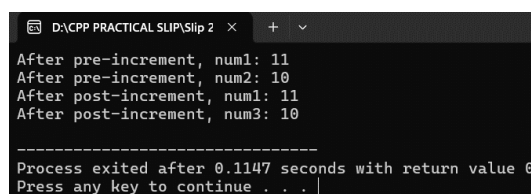
```

Slip 20

A) Create A C++ Class Number With Integer Data Member. Write Necessary Member Functions To Overload The Operator Unary Pre And Post Increment '++'...

```
#include <iostream>
class Number {
private:
    int num;
public:
    Number(int n = 0) : num(n) {}
    Number& operator++() {
        ++num;
        return *this; }
    Number operator++(int) {
        Number temp(num);
        num++;
        return temp; }
    int getNum() const {
        return num;
    }
};

int main() {
    Number num1(9);
    Number num2 = ++num1;
    Number num3 = num1++;
    std::cout << "After pre-increment, num1: " << num1.getNum()
    << std::endl;
    std::cout << "After pre-increment, num2: " << num2.getNum()
    << std::endl;
    std::cout << "After post-increment, num1: " << num1.getNum()
    << std::endl;
    std::cout << "After post-increment, num3: " << num3.getNum()
    << std::endl;
    return 0;
}
```

A screenshot of a terminal window showing the output of a C++ program. The window title is 'D:\CPP PRACTICAL SLIP\Slip 2 x'. The output text is: 'After pre-increment, num1: 11', 'After pre-increment, num2: 10', 'After post-increment, num1: 11', 'After post-increment, num3: 10'. Below this, a separator line is shown, followed by 'Process exited after 0.1147 seconds with return value 0' and 'Press any key to continue . . . |'.

B) Create A C++ Class For Inventory Of Mobiles With Data Members Model, Mobile _Company, Color, Price And Quantity. Mobile Can Be Sold, If Stock Is Available, Otherwise Purchase Will Be Made. Write Necessary Member Functions For The Following:

- **To Accept Mobile Details From User.**
- **To Sale A Mobile. (Sale Contains Mobile Details & Number Of Mobiles To Be Sold.)**
- **To Purchase A Mobile. (Purchase Contains Mobile Details & Number Of Mobiles To Be Purchased)**

```
#include <iostream>
#include <string>
using namespace std;
class MobileInventory {
private:
    string model;
    string company;
    string color;
    double price;
    int quantity;
public:
    MobileInventory() : model(""), company(""), color(""), price(0.0),
quantity(0) {}
    void acceptDetails() {
        cout << "Enter model: ";
        getline(cin, model);
        cout << "Enter company: ";
        getline(cin, company);
        cout << "Enter color: ";
        getline(cin, color);
        cout << "Enter price: ";
        cin >> price;
        cout << "Enter quantity: ";
        cin >> quantity;
        cin.ignore(); // Clear the input buffer
    }
    void sale(int numOfMobiles) {
```



```

        if (quantity >= numOfMobiles) {
            cout << numOfMobiles << " mobile(s) sold.\n";
            quantity -= numOfMobiles;
        } else {
            cout << "Sorry, only " << quantity << " mobile(s) available
for sale.\n";
        }
    }
}

void purchase(int numOfMobiles) {
    cout << "Purchased " << numOfMobiles << " mobile(s).\n";
    quantity += numOfMobiles;
}

void display() {
    cout << "Model: " << model << endl;
    cout << "Company: " << company << endl;
    cout << "Color: " << color << endl;
    cout << "Price: " << price << endl;
    cout << "Quantity: " << quantity << endl;
}

};

int main() {
    MobileInventory inventory;
    inventory.acceptDetails();
    cout << "\nMobile Details:\n";
    inventory.display();
    inventory.sale(2);
    cout << "\nUpdated Mobile Details:\n";
    inventory.display();
    inventory.purchase(5);
    cout << "\nUpdated Mobile Details:\n";
    inventory.display();
    return 0;
}

```

The screenshot shows a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 2" with the following output:

```

Enter model: Vivo T2 5G
Enter company: Vivo
Enter color: Blue
Enter price: 18000
Enter quantity: 10

Mobile Details:
Model: Vivo T2 5G
Company: Vivo
Color: Blue
Price: 18000
Quantity: 10
2 mobile(s) sold.

Updated Mobile Details:
Model: Vivo T2 5G
Company: Vivo
Color: Blue
Price: 18000
Quantity: 8
Purchased 5 mobile(s).

Updated Mobile Details:
Model: Vivo T2 5G
Company: Vivo
Color: Blue
Price: 18000
Quantity: 13

Process exited after 55.63 seconds with return value 0
Press any key to continue . . .

```

Slip 21

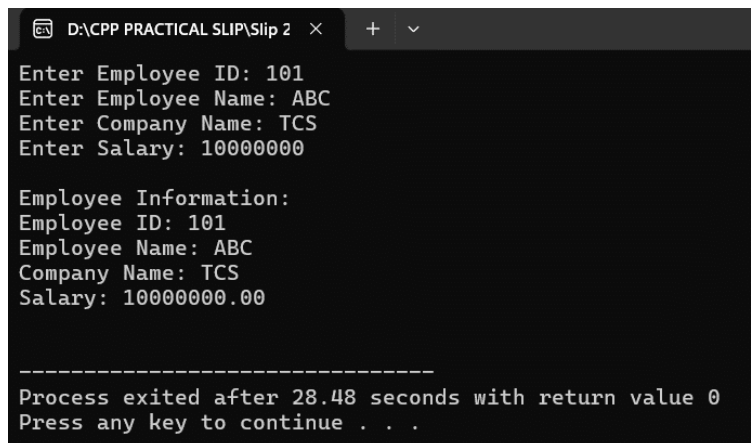
A) Create A C++ Class Employee With Data Members Emp_Id, Emp_Name, Company_Name And Salary. Write Member Functions To Accept And Display Employee Information. Design User Defined Manipulator To Print Salary.(For Salary Set Right Justification, Maximum Width To 7 And Fill Remaining Spaces With ‘*’) ...

```
#include <iostream>
#include <iomanip>
#include <string>
class Employee {
private:
    int emp_id;
    std::string emp_name;
    std::string company_name;
    double salary;
public:
    Employee(int id, const std::string& name, const std::string&
company, double sal)
        : emp_id(id), emp_name(name), company_name(company),
salary(sal) {}
    void acceptInfo() {
        std::cout << "Enter Employee ID: ";
        std::cin >> emp_id;
        std::cin.ignore(); // Clear input buffer
        std::cout << "Enter Employee Name: ";
        std::getline(std::cin, emp_name);
        std::cout << "Enter Company Name: ";
        std::getline(std::cin, company_name);
        std::cout << "Enter Salary: ";
        std::cin >> salary;
    }
    void displayInfo() const {
        std::cout << "Employee ID: " << emp_id << std::endl;
        std::cout << "Employee Name: " << emp_name << std::endl;
        std::cout << "Company Name: " << company_name <<
std::endl;
        std::cout << "Salary: ";
```

```

    }
    friend std::ostream& operator<<(std::ostream& os, const
Employee& emp);
};
std::ostream& salaryFormat(std::ostream& os) {
    os << std::setw(7) << std::right << std::fixed <<
std::setprecision(2) << std::setfill('*');
    return os;
}
std::ostream& operator<<(std::ostream& os, const Employee& emp)
{
    emp.displayInfo();
    os << salaryFormat << emp.salary << std::endl;
    return os;
}
int main() {
    Employee emp(0, "", "", 0.0);
    emp.acceptInfo();
    std::cout << "\nEmployee Information:\n" << emp << std::endl;
    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 2". The program prompts for employee information: "Enter Employee ID: 101", "Enter Employee Name: ABC", "Enter Company Name: TCS", and "Enter Salary: 10000000". It then displays the "Employee Information:" with the entered details: "Employee ID: 101", "Employee Name: ABC", "Company Name: TCS", and "Salary: 10000000.00". The output is formatted with right-aligned numbers and fixed precision. At the bottom, it shows "Process exited after 28.48 seconds with return value 0" and "Press any key to continue . . .".

```

D:\CPP PRACTICAL SLIP\Slip 2
Enter Employee ID: 101
Enter Employee Name: ABC
Enter Company Name: TCS
Enter Salary: 10000000

Employee Information:
Employee ID: 101
Employee Name: ABC
Company Name: TCS
Salary: 10000000.00

-----
Process exited after 28.48 seconds with return value 0
Press any key to continue . . .

```

B) Create A C++ Class For A Two Dimensional Points. Write Necessary Member Functions To Accept & Display The Point Object. Overload The Following Operators:

Operator	Example	Purpose
+(Binary)	$P_3 = P_1 + P_2$	Adds Coordinates of point p1 and p2.
-(Unary)	$-P_1$	Negates Coordinates of point p1.
*(Binary)	$P_2 = P_1 * n$	Multiply Coordinates of point p1 by constant 'n'

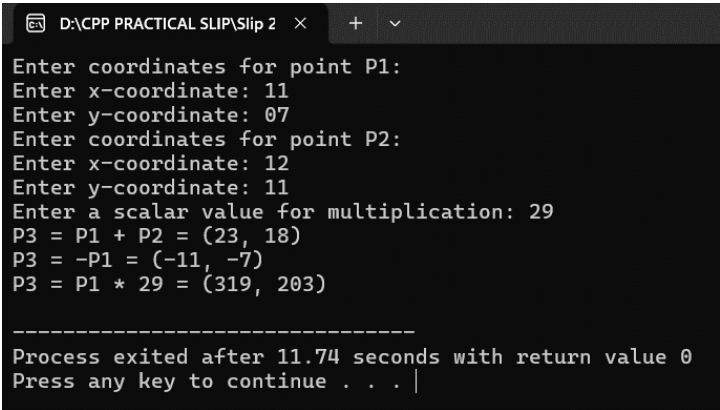
```
#include <iostream>
class Point2D {
private:
    double x, y;
public:
    Point2D(double x_val = 0.0, double y_val = 0.0) : x(x_val),
y(y_val) {}
    void acceptPoint() {
        std::cout << "Enter x-coordinate: ";
        std::cin >> x;
        std::cout << "Enter y-coordinate: ";
        std::cin >> y;
    }
    void displayPoint() const {
        std::cout << "(" << x << ", " << y << ")";
    }
    Point2D operator+(const Point2D& rhs) const {
        return Point2D(x + rhs.x, y + rhs.y);
    }
    Point2D operator-() const {
        return Point2D(-x, -y);
    }
    Point2D operator*(double scalar) const {
        return Point2D(x * scalar, y * scalar);
    }
};

int main() {
    Point2D P1, P2, P3;
    double n;
```

```

std::cout << "Enter coordinates for point P1:\n";
P1.acceptPoint();
std::cout << "Enter coordinates for point P2:\n";
P2.acceptPoint();
std::cout << "Enter a scalar value for multiplication: ";
std::cin >> n;
P3 = P1 + P2;
std::cout << "P3 = P1 + P2 = ";
P3.displayPoint();
std::cout << std::endl;
P3 = -P1;
std::cout << "P3 = -P1 = ";
P3.displayPoint();
std::cout << std::endl;
P3 = P1 * n;
std::cout << "P3 = P1 * " << n << " = ";
P3.displayPoint();
std::cout << std::endl;
return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 2
Enter coordinates for point P1:
Enter x-coordinate: 11
Enter y-coordinate: 07
Enter coordinates for point P2:
Enter x-coordinate: 12
Enter y-coordinate: 11
Enter a scalar value for multiplication: 29
P3 = P1 + P2 = (23, 18)
P3 = -P1 = (-11, -7)
P3 = P1 * 29 = (319, 203)

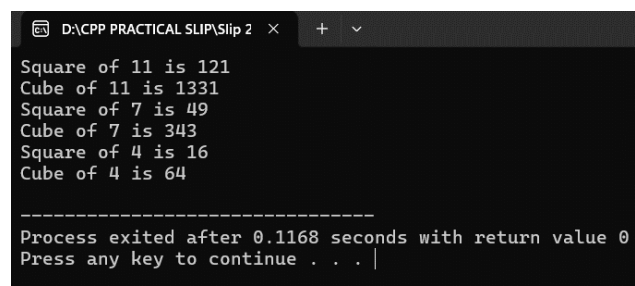
-----
Process exited after 11.74 seconds with return value 0
Press any key to continue . . .

```

Slip 22

A) Write A C++ Program To Define Two Function Templates For Calculating The Square And Cube Of Given Numbers With Different Data Types...

```
#include <iostream>
template<typename T>
T square(T num) {
    return num * num;
}
template<typename T>
T cube(T num) {
    return num * num * num;
}
int main() {
    int intNum = 11;
    float floatNum = 07;
    double doubleNum = 04;
    std::cout << "Square of " << intNum << " is " <<
square(intNum) << std::endl;
    std::cout << "Cube of " << intNum << " is " << cube(intNum)
<< std::endl;
    std::cout << "Square of " << floatNum << " is " <<
square(floatNum) << std::endl;
    std::cout << "Cube of " << floatNum << " is " << cube(floatNum)
<< std::endl;
    std::cout << "Square of " << doubleNum << " is " <<
square(doubleNum) << std::endl;
    std::cout << "Cube of " << doubleNum << " is " <<
cube(doubleNum) << std::endl;
    return 0;
}
```

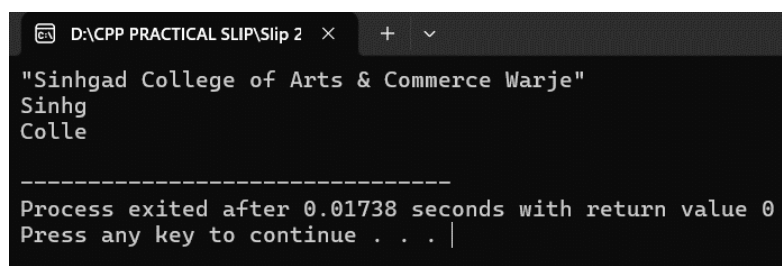
A screenshot of a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 2". The window shows the output of the C++ program, displaying the square and cube of integers 11, 7, and 4, as well as floats 07 and 04. The output is as follows:
Square of 11 is 121
Cube of 11 is 1331
Square of 7 is 49
Cube of 7 is 343
Square of 4 is 16
Cube of 4 is 64

Process exited after 0.1168 seconds with return value 0
Press any key to continue . . .|

B) Write A C++ Program To Overload 'Display_Str' Function As Follows:

- **void display_str(char *)** - Display a string in double quotes.
- **void display_str (int n, char *)**- Display first n characters from a given string.
- **void display_str (int m, int n, char *)**- Display substring of a given string from position m to n.

```
#include <iostream>
#include <cstring>
void display_str(char *str) {
    std::cout << "\"" << str << "\"" << std::endl;
}
void display_str(int n, char *str) {
    for (int i = 0; i < n && str[i] != '\0'; ++i) {
        std::cout << str[i];
    }
    std::cout << std::endl;
}
void display_str(int m, int n, char *str) {
    int len = strlen(str);
    if (m < 0 || m >= len || n < 0 || n >= len || m > n) {
        std::cout << "Invalid range!" << std::endl;
        return;
    }
    for (int i = m; i <= n; ++i) {
        std::cout << str[i];
    }
    std::cout << std::endl;
}
int main() {
    char str[] = "Sinhgad College of Arts & Commerce Warje";
    display_str(str);
    display_str(5, str);
    display_str(7, 12, str);
    return 0;
}
```



The screenshot shows a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 2". The output of the program is displayed as follows:

```
"Sinhgad College of Arts & Commerce Warje"
Sinhg
Colle
-----
Process exited after 0.01738 seconds with return value 0
Press any key to continue . . . |
```

Slip 23

A) Create A C++ Class Mystring With Data Member Character Pointer. Write A C++ Program To Accept And Display A String. Overload '+' Operator To Concatenate Two Strings...

```
#include <iostream>
#include <cstring>
class MyString {
private:
    char *str;
public:
    MyString(const char *s = NULL) {
        if (s == NULL) {
            str = new char[1];
            *str = '\0';
        } else {
            str = new char[strlen(s) + 1];
            strcpy(str, s);
        }
    }
    ~MyString() {
        delete[] str;
    }
    MyString(const MyString &source) {
        str = new char[strlen(source.str) + 1];
        strcpy(str, source.str);
    }
    MyString &operator=(const MyString &source) {
        if (this == &source)
            return *this;
        delete[] str;
        str = new char[strlen(source.str) + 1];
        strcpy(str, source.str);
        return *this;
    }
    void display() const {
        std::cout << str << std::endl;
    }
    MyString operator+(const MyString &other) const {
```

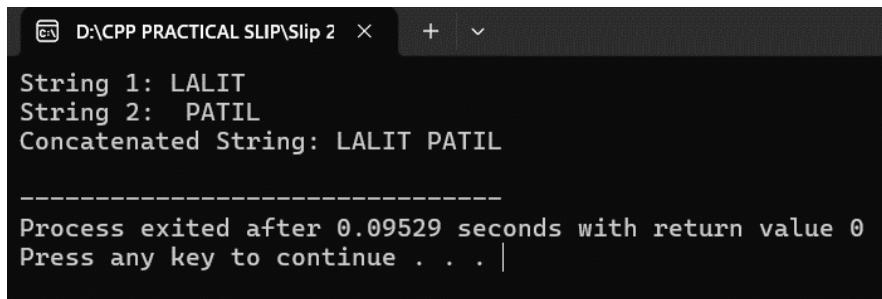


```

        char *temp = new char[strlen(str) + strlen(other.str) + 1];
        strcpy(temp, str);
        strcat(temp, other.str);
        MyString result(temp);
        delete[] temp;
        return result;
    }
};

int main() {
    MyString str1("LALIT");
    MyString str2(" PATIL");
    MyString result = str1 + str2;
    std::cout << "String 1: ";
    str1.display();
    std::cout << "String 2: ";
    str2.display();
    std::cout << "Concatenated String: ";
    result.display();
    return 0;
}

```



The screenshot shows a terminal window with the title "D:\CPP PRACTICAL SLIP\Slip 2". The output of the program is as follows:

```

String 1: LALIT
String 2:  PATIL
Concatenated String: LALIT PATIL

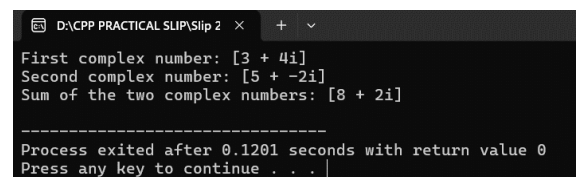
-----
Process exited after 0.09529 seconds with return value 0
Press any key to continue . . . |

```

B) Create A C++ Class Complexnumber With Data Members Real And Imaginary. Write Necessary Functions:

- i. To accept Complex Number using constructor.**
- ii. To display Complex Number in format $[x + iy]$.**
- iii. To add two Complex Numbers by using friend function...**

```
#include <iostream>
class ComplexNumber {
private:
    double real;
    double imaginary;
public:
    ComplexNumber(double realPart, double imaginaryPart) {
        real = realPart;
        imaginary = imaginaryPart; }
    void display() const {
        std::cout << "[" << real << " + " << imaginary << "i]"; }
    friend ComplexNumber add(const ComplexNumber& num1, const
ComplexNumber& num2);};
ComplexNumber add(const ComplexNumber& num1, const
ComplexNumber& num2) {
    double realSum = num1.real + num2.real;
    double imaginarySum = num1.imaginary + num2.imaginary;
    return ComplexNumber(realSum, imaginarySum);}
int main() {
    ComplexNumber num1(3, 4);
    ComplexNumber num2(5, -2);
    std::cout << "First complex number: ";
    num1.display();
    std::cout << std::endl;
    std::cout << "Second complex number: ";
    num2.display();
    std::cout << std::endl;
    ComplexNumber sum = add(num1, num2);
    std::cout << "Sum of the two complex numbers: ";
    sum.display(); std::cout << std::endl;
    return 0;}
```



The screenshot shows a terminal window with the title "D:\CPP PRACTICAL SLIP\Slip 2". The output of the program is as follows:

```
First complex number: [3 + 4i]
Second complex number: [5 + -2i]
Sum of the two complex numbers: [8 + 2i]

-----
Process exited after 0.1201 seconds with return value 0
Press any key to continue . . .
```

Slip 24

A) Create a C++ class Fix Deposit with data members FD_No, Cust_Name, FD_Amt, Interest rate, Maturity amt, Number_of_months. Create and Initialize all values of FixDeposit object by using parameterized constructor with default value for interest rate. Calculate maturity amt using interest rate and display all the details.

```
#include <iostream>
#include <string>
#include <iomanip>
class FixDeposit {
private:
    int FD_No;
    std::string Cust_Name;
    double FD_Amt;
    double Interest_Rate;
    double Maturity_Amt;
    int Number_of_Months;
public:
    FixDeposit(int fdNo, const std::string& custName, double fdAmt,
int numOfMonths, double interestRate = 0.05) :
        FD_No(fdNo),Cust_Name(custName),FD_Amt(fdAmt),
Interest_Rate(interestRate), Number_of_Months(numOfMonths) {
        calculateMaturityAmt();
    }
    void calculateMaturityAmt() {
        double r = Interest_Rate / 100;
        Maturity_Amt=FD_Amt+(FD_Amt * r *
Number_of_Months);
    }
    void displayDetails() {
        std::cout << "Fixed Deposit Details:" << std::endl;
        std::cout << "FD Number: " << FD_No << std::endl;
        std::cout << "Customer Name: " << Cust_Name << std::endl;
        std::cout << "FD Amount: $" << std::fixed <<
std::setprecision(2) << FD_Amt << std::endl;
        std::cout << "Interest Rate: " << Interest_Rate << "%" <<
std::endl;
```

```

        std::cout << "Number of Months: " << Number_of_Months <<
std::endl;
        std::cout << "Maturity Amount: $" << std::fixed <<
std::setprecision(2) << Maturity_Amt << std::endl;
    }
};
int main() {
    FixDeposit fd1(12345, "ABC", 5000.00, 12);
    fd1.displayDetails();
    std::cout << std::endl;
    FixDeposit fd2(54321, "PQR", 10000.00, 24, 6.5);
    fd2.displayDetails();
    return 0;
}

```

```

D:\CPP PRACTICAL SLIP\Slip 2
Fixed Deposit Details:
FD Number: 12345
Customer Name: ABC
FD Amount: $5000.00
Interest Rate: 0.05%
Number of Months: 12
Maturity Amount: $5030.00

Fixed Deposit Details:
FD Number: 54321
Customer Name: PQR
FD Amount: $10000.00
Interest Rate: 6.50%
Number of Months: 24
Maturity Amount: $25600.00

-----
Process exited after 0.1318 seconds with return value 0
Press any key to continue . . .

```

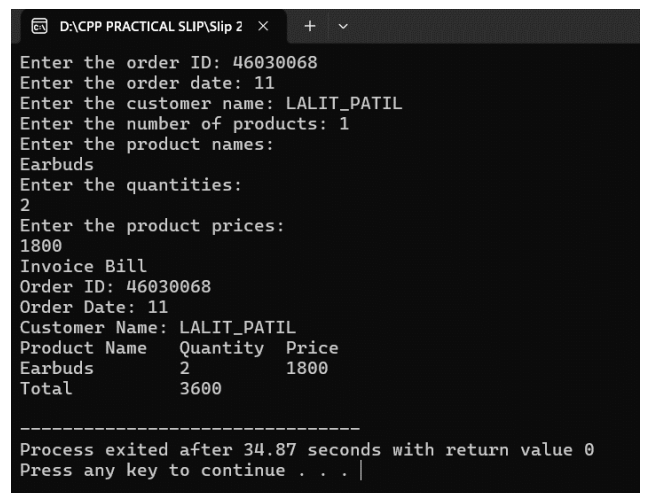
B) Create A C++ Class Invoicebill With Data Members Order Id, O Date, Customer Name, No_Of_Products, Prod_Name[], Quantity], Prod_Price[]. A Customer Can Buy 'N' Products. Accept Quantity For Each Product. Generate And Display The Bill Using Appropriate Manipulators. (Use New Operator)...

```
#include <iostream>
#include <iomanip>
using namespace std;
class InvoiceBill {
private:
    int Order_id;
    string O_Date;
    string Customer_Name;
    int No_of_Products;
    string* Prod_Name;
    int* Quantity;
    float* Prod_Price;
public:
    InvoiceBill(int Order_id, string O_Date, string Customer_Name, int
No_of_Products) { this->Order_id = Order_id;
        this->O_Date = O_Date;
        this->Customer_Name = Customer_Name;
        this->No_of_Products = No_of_Products;
        Prod_Name = new string[No_of_Products];
        Quantity = new int[No_of_Products];
        Prod_Price = new float[No_of_Products]; }
    ~InvoiceBill() {
        delete[] Prod_Name;
        delete[] Quantity;
        delete[] Prod_Price;}
    void acceptDetails() {
        cout << "Enter the product names: " << endl;
        for (int i = 0; i < No_of_Products; i++) {
            cin >> Prod_Name[i]; }
        cout << "Enter the quantities: " << endl;
        for (int i = 0; i < No_of_Products; i++) {
            cin >> Quantity[i]; }
        cout << "Enter the product prices: " << endl;
```

```

    for (int i = 0; i < No_of_Products; i++) {
        cin >> Prod_Price[i]; } }
void generateBill() {
    cout << "Invoice Bill" << endl;
    cout << "Order ID: " << Order_id << endl;
    cout << "Order Date: " << O_Date << endl;
    cout << "Customer Name: " << Customer_Name << endl;
    cout << setw(15) << left << "Product Name" << setw(10) << left
    << "Quantity" << setw(10) << left << "Price" << endl;
    for (int i = 0; i < No_of_Products; i++) {
        cout << setw(15) << left << Prod_Name[i] << setw(10) << left
        << Quantity[i] << setw(10) << left << Prod_Price[i] << endl; }
    float total = 0;
    for (int i = 0; i < No_of_Products; i++) {
        total += Quantity[i] * Prod_Price[i]; }
    cout << setw(15) << left << "Total" << setw(10) << left << total
    << endl; } };
int main() {
    int Order_id, No_of_Products;
    string O_Date, Customer_Name;
    cout << "Enter the order ID: ";
    cin >> Order_id;
    cout << "Enter the order date: ";
    cin >> O_Date;
    cout << "Enter the customer name: ";
    cin >> Customer_Name;
    cout << "Enter the number of products: ";
    cin >> No_of_Products;
    InvoiceBill* bill = new InvoiceBill(Order_id, O_Date, Customer_Name,
    No_of_Products);
    bill->acceptDetails();
    bill->generateBill();
    delete bill;
    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 2 x + v
Enter the order ID: 46030068
Enter the order date: 11
Enter the customer name: LALITPATIL
Enter the number of products: 1
Enter the product names:
Earbuds
Enter the quantities:
2
Enter the product prices:
1800
Invoice Bill
Order ID: 46030068
Order Date: 11
Customer Name: LALITPATIL
Product Name  Quantity  Price
Earbuds        2         1800
Total          3600

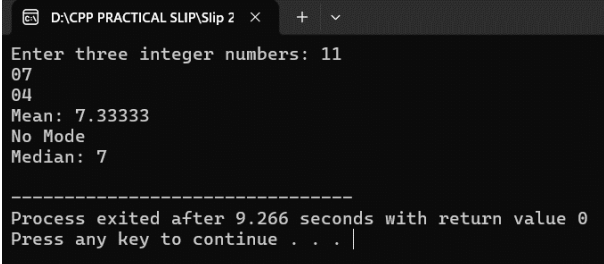
-----
Process exited after 34.87 seconds with return value 0
Press any key to continue . . .

```

Slip 25

A) Write A C++ Program To Calculate Mean, Mode And Median Of Three Integer Numbers. (Use Inline Function)

```
#include <iostream>
#include <algorithm>
using namespace std;
inline float Mean(int a, int b, int c) {
    return (a + b + c) / 3.0;}
inline int Mode(int a, int b, int c) {
    if (a == b && b == c)
        return a;
    else if (a == b || b == c)
        return b;
    else if (a == c)
        return a;
    else
        return -1; }
inline float Median(int a, int b, int c) {
    int arr[3] = {a, b, c};
    sort(arr, arr + 3);
    return arr[1];}
int main() {
    int num1, num2, num3;
    cout << "Enter three integer numbers: ";
    cin >> num1 >> num2 >> num3;
    cout << "Mean: " << Mean(num1, num2, num3) << endl;
    int mode = Mode(num1, num2, num3);
    if (mode != -1)
        cout << "Mode: " << mode << endl; else
        cout << "No Mode" << endl;
    cout << "Median: " << Median(num1, num2, num3) << endl;
    return 0;}
```



The screenshot shows a Windows command prompt window titled "D:\C++ PRACTICAL SLIP\Slip 2 x". The program prompts the user to "Enter three integer numbers: 11 07 04". It then displays the results: "Mean: 7.33333", "No Mode", and "Median: 7". At the bottom, it shows "Process exited after 9.266 seconds with return value 0" and "Press any key to continue . . . |".

B) Write A C++ Program To Read The Contents Of A "Sampledata.Txt" File. Create "Upper.Txt" File To Store Uppercase Characters, "Lower.Txt" File To Store Lowercase Characters, "Digit.Txt" File To Store Digits And "Other.Txt" File To Store Remaining Characters Of A Given File.

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
int main() {
    ifstream inputFile("SampleData.txt");
    ofstream upperFile("Upper.txt");
    ofstream lowerFile("Lower.txt");
    ofstream digitFile("Digit.txt");
    ofstream otherFile("Other.txt");
    if (!inputFile) {
        cerr << "Error: Unable to open input file.\n";
        return 1;
    }
    if (!upperFile || !lowerFile || !digitFile || !otherFile) {
        cerr << "Error: Unable to create output files.\n";
        return 1;
    }
    char ch;
    while (inputFile.get(ch)) {
        if (isupper(ch))
            upperFile.put(ch);
        else if (islower(ch))
            lowerFile.put(ch);
        else if (isdigit(ch))
            digitFile.put(ch);
        else
            otherFile.put(ch);
    }
    inputFile.close();
    upperFile.close();
    lowerFile.close();
    digitFile.close();
}
```



```

otherFile.close();
cout << "Files created successfully.\n";
return 0;
}

```

```

D:\CPP PRACTICAL SLIP\Slip 2
Files created successfully.
-----
Process exited after 0.1346 seconds with return value 0
Press any key to continue . . .

```

Slip 25 B.cpp	SampleData.txt	Lower.txt	Other.txt	Upper.txt
1	!@#\$\$%^&*()_+	1	LALITPATIL	
2	LALITPATIL			
3	lalit patil			
4	12112920			
5	!@#\$\$%^&*()_+			

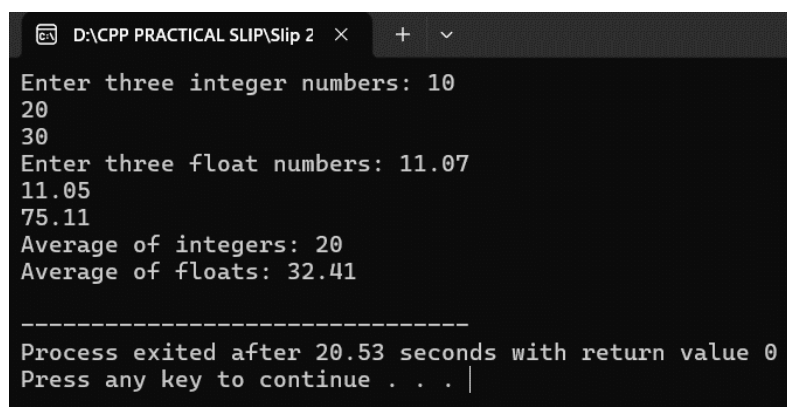
Slip 25 B.cpp	SampleData.txt	Lower.txt	Other.txt	Upper.txt
1	!@#\$\$%^&*()_+	1	lalitpatil	
2	LALITPATIL			
3	lalit patil			
4	12112920			
5	!@#\$\$%^&*()_+			

Slip 25 B.cpp	SampleData.txt	Lower.txt	Other.txt	Upper.txt
1	!@#\$\$%^&*()_+	1	!@#\$\$%^&*()_+	
2	LALITPATIL			
3	lalit patil			
4	12112920			
5	!@#\$\$%^&*()_+			

Slip 26

A) Write A C++ Program To Find Average Of 3 Integer Numbers And Average Of 3 Float Numbers. (Use Function Overloading)...

```
#include <iostream>
using namespace std;
float average(int num1, int num2, int num3) {
    return static_cast<float>(num1 + num2 + num3) / 3;
}
float average(float num1, float num2, float num3) {
    return (num1 + num2 + num3) / 3.0f;
}
int main() {
    int int1, int2, int3;
    float float1, float2, float3;
    cout << "Enter three integer numbers: ";
    cin >> int1 >> int2 >> int3;
    cout << "Enter three float numbers: ";
    cin >> float1 >> float2 >> float3;
    cout << "Average of integers: " << average(int1, int2, int3) <<
endl;
    cout << "Average of floats: " << average(float1, float2, float3) <<
endl;
    return 0;
}
```



The screenshot shows a Windows command prompt window titled "D:\C++ PRACTICAL SLIP\Slip 2". The program prompts for three integer numbers (10, 20, 30) and three float numbers (11.07, 11.05, 75.11). It then displays the calculated averages: 20 for integers and 32.41 for floats. At the bottom, it shows "Process exited after 20.53 seconds with return value 0" and "Press any key to continue . . .".

```
D:\C++ PRACTICAL SLIP\Slip 2
Enter three integer numbers: 10
20
30
Enter three float numbers: 11.07
11.05
75.11
Average of integers: 20
Average of floats: 32.41

-----
Process exited after 20.53 seconds with return value 0
Press any key to continue . . .
```

B) Create A C++ Class Time With Data Members Hours, Minutes, Seconds. Write A C++ Program Using operator Overloading To Perform The Following:

- i. ! To check whether two Times are equal or not.**
- ii. >> To accept the time.**
- iii. << To display the time.**

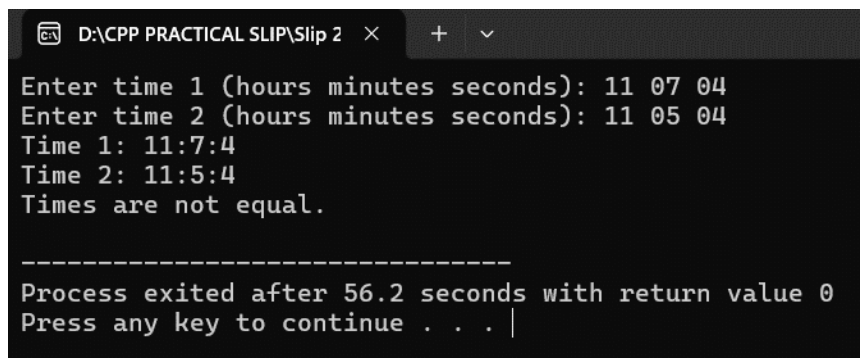
```
#include <iostream>
class Time {
private:
    int hours;
    int minutes;
    int seconds;
public:
    Time() : hours(0), minutes(0), seconds(0) {}
    Time(int h, int m, int s) : hours(h), minutes(m), seconds(s) {}
    friend std::istream& operator>>(std::istream& in, Time& t);
    friend std::ostream& operator<<(std::ostream& out, const Time&
t);
    friend bool operator==(const Time& t1, const Time& t2);
};
std::istream& operator>>(std::istream& in, Time& t) {
    in >> t.hours >> t.minutes >> t.seconds;
    return in;
}
std::ostream& operator<<(std::ostream& out, const Time& t) {
    out << t.hours << ":" << t.minutes << ":" << t.seconds;
    return out;
}
bool operator==(const Time& t1, const Time& t2) {
    return (t1.hours == t2.hours && t1.minutes == t2.minutes &&
t1.seconds == t2.seconds);
}
int main() {
    Time t1, t2;
    std::cout << "Enter time 1 (hours minutes seconds): ";
    std::cin >> t1;
    std::cout << "Enter time 2 (hours minutes seconds): ";
    std::cin >> t2;
```

```

std::cout << "Time 1: " << t1 << std::endl;
std::cout << "Time 2: " << t2 << std::endl;
if (t1 == t2)
    std::cout << "Both times are equal." << std::endl;
else
    std::cout << "Times are not equal." << std::endl;

return 0;
}

```



The screenshot shows a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 2". The program prompts the user to enter two times in "hours minutes seconds" format. The first input is "11 07 04" and the second is "11 05 04". The program then displays the entered times as "Time 1: 11:7:4" and "Time 2: 11:5:4". Since the times are not equal, it outputs "Times are not equal.". Below this, a separator line is shown, followed by the message "Process exited after 56.2 seconds with return value 0" and a prompt "Press any key to continue . . . |".

```

D:\CPP PRACTICAL SLIP\Slip 2
Enter time 1 (hours minutes seconds): 11 07 04
Enter time 2 (hours minutes seconds): 11 05 04
Time 1: 11:7:4
Time 2: 11:5:4
Times are not equal.

-----
Process exited after 56.2 seconds with return value 0
Press any key to continue . . . |

```

Slip 27

A) Create A C++ Class College, With Data Members College_Id, College_Name, Establishment_Year, University_Name. Overload Operators >> And << To Accept And Display College Information.

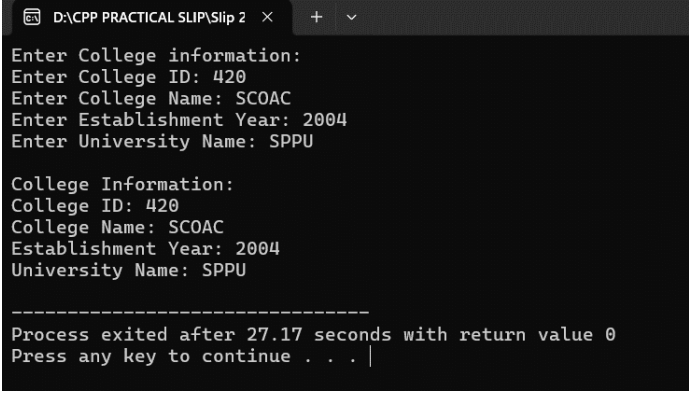
```
#include <iostream>
#include <string>
class College {
private:
    int collegeId;
    std::string collegeName;
    int establishmentYear;
    std::string universityName;
public:
    College() : collegeId(0), establishmentYear(0) {}
    College(int id, const std::string& name, int year, const std::string&
university)
        : collegeId(id), collegeName(name), establishmentYear(year),
universityName(university) {}
    friend std::ostream& operator<<(std::ostream& os, const College&
college) {
        os << "College ID: " << college.collegeId << std::endl;
        os << "College Name: " << college.collegeName << std::endl;
        os << "Establishment Year: " << college.establishmentYear
<< std::endl;
        os << "University Name: " << college.universityName <<
std::endl;
        return os;
    }
    friend std::istream& operator>>(std::istream& is, College&
college) {
        std::cout << "Enter College ID: ";
        is >> college.collegeId;
        std::cout << "Enter College Name: ";
        std::cin.ignore();
        std::getline(is, college.collegeName);
        std::cout << "Enter Establishment Year: ";
        is >> college.establishmentYear;
```

```

        std::cout << "Enter University Name: ";
        std::cin.ignore();
        std::getline(is, college.universityName);
        return is;
    }
};

int main() {
    College college;
    std::cout << "Enter College information: " << std::endl;
    std::cin >> college;
    std::cout << "\nCollege Information:" << std::endl;
    std::cout << college;
    return 0;
}

```



```

D:\CPP PRACTICAL SLIP\Slip 2 >
Enter College information:
Enter College ID: 420
Enter College Name: SCOAC
Enter Establishment Year: 2004
Enter University Name: SPPU

College Information:
College ID: 420
College Name: SCOAC
Establishment Year: 2004
University Name: SPPU

-----
Process exited after 27.17 seconds with return value 0
Press any key to continue . . .

```

B) Create A Base Class Travels With Data Members T_No, Company Name. Derive A Class Route With Data Members Route Id, Source, And Destination From Travels Class. Also Derive A Class Reservation With Data Members Number Of Seats, Travels Class, Fare, And Travel Date From Route.

Write a C++ program to perform following necessary member functions:

- i. Accept details of 'n' reservations.**
- ii. Display details of all reservations.**
- iii. Display reservation details of a specified Date.**

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class Travels {
protected:
    int T_No;
    string CompanyName;
public:
    void acceptDetails() {
        cout << "Enter Travel Number: ";
        cin >> T_No;
        cout << "Enter Company Name: ";
        cin.ignore();
        getline(cin, CompanyName);
    }
    void displayDetails() {
        cout << "Travel Number: " << T_No << endl;
        cout << "Company Name: " << CompanyName << endl;
    }
};
class Route : public Travels {
private:
    int RouteId;
    string Source;
    string Destination;
public:
    void acceptDetails() {
```

```

    Travels::acceptDetails();
    cout << "Enter Route ID: ";
    cin >> RouteId;
    cout << "Enter Source: ";
    cin.ignore(); // clear input buffer
    getline(cin, Source);
    cout << "Enter Destination: ";
    getline(cin, Destination);
}
void displayDetails() {
    Travels::displayDetails();
    cout << "Route ID: " << RouteId << endl;
    cout << "Source: " << Source << endl;
    cout << "Destination: " << Destination << endl;
}
};
class Reservation : public Route {
private:
    int NumSeats;
    float Fare;
    string TravelDate;
public:
    void acceptDetails() {
        Route::acceptDetails();
        cout << "Enter Number of Seats: ";
        cin >> NumSeats;
        cout << "Enter Fare: ";
        cin >> Fare;
        cout << "Enter Travel Date: ";
        cin.ignore(); // clear input buffer
        getline(cin, TravelDate);
    }
    void displayDetails() {
        Route::displayDetails();
        cout << "Number of Seats: " << NumSeats << endl;
        cout << "Fare: " << Fare << endl;
        cout << "Travel Date: " << TravelDate << endl;
    }
    string getTravelDate() const {

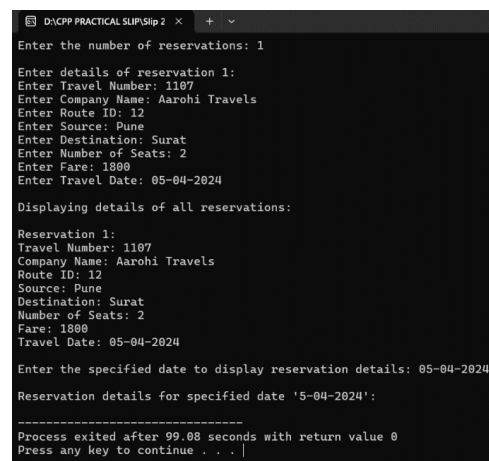
```



```

        return TravelDate;
    }
};
int main() {
    int n;
    cout << "Enter the number of reservations: ";
    cin >> n;
    vector<Reservation> reservations(n);
    for (int i = 0; i < n; ++i) {
        cout << "\nEnter details of reservation " << i + 1 << ": " <<
endl;
        reservations[i].acceptDetails();
    }
    cout << "\nDisplaying details of all reservations:" << endl;
    for (int i = 0; i < n; ++i) {
        cout << "\nReservation " << i + 1 << ": " << endl;
        reservations[i].displayDetails();
    }
    string specifiedDate;
    cout << "\nEnter the specified date to display reservation details:
";
    cin.ignore();
    getline(cin, specifiedDate);
    cout << "\nReservation details for specified date " <<
specifiedDate << ": " << endl;
    for (int i = 0; i < n; ++i) {
        if (reservations[i].getTravelDate() == specifiedDate) {
            reservations[i].displayDetails();
        }
    }
    return 0;
}

```



```

D:\C++ PRACTICALS\Slip2 x + v
Enter the number of reservations: 1
Enter details of reservation 1:
Enter Travel Number: 1107
Enter Company Name: Aarohi Travels
Enter Route ID: 12
Enter Source: Pune
Enter Destination: Surat
Enter Number of Seats: 2
Enter Fare: 1800
Enter Travel Date: 05-04-2024

Displaying details of all reservations:

Reservation 1:
Travel Number: 1107
Company Name: Aarohi Travels
Route ID: 12
Source: Pune
Destination: Surat
Number of Seats: 2
Fare: 1800
Travel Date: 05-04-2024

Enter the specified date to display reservation details: 05-04-2024

Reservation details for specified date '5-04-2024':

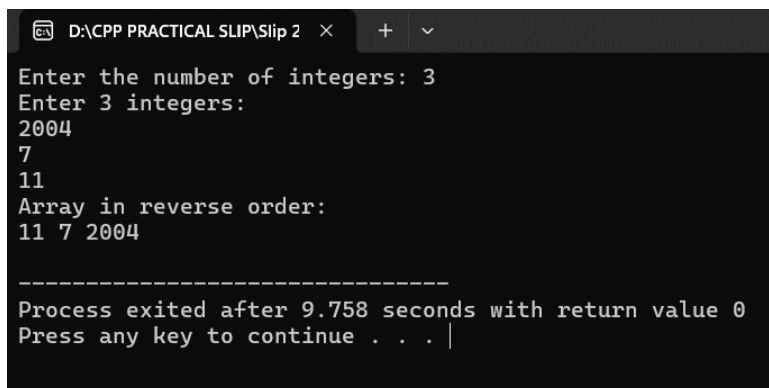
-----
Process exited after 99.08 seconds with return value 0
Press any key to continue . . . |

```

Slip 28

A) Write A C++ Program To Read Array Of 'N' Integers From User And Display It In Reverse Order. (Use Dynamic Memory Allocation).

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter the number of integers: ";
    cin >> n;
    int* arr = new int[n];
    cout << "Enter " << n << " integers:" << endl;
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }
    cout << "Array in reverse order:" << endl;
    for (int i = n - 1; i >= 0; --i) {
        cout << arr[i] << " ";
    }
    cout << endl;
    delete[] arr;
    return 0;
}
```

A screenshot of a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 2". The window shows the execution of a C++ program. The user enters "3" for the number of integers, then "2004", "7", and "11" for the integers. The program outputs "Array in reverse order:" followed by "11 7 2004". At the bottom, it says "Process exited after 9.758 seconds with return value 0" and "Press any key to continue . . .".

```
D:\CPP PRACTICAL SLIP\Slip 2  x  +  v
Enter the number of integers: 3
Enter 3 integers:
2004
7
11
Array in reverse order:
11 7 2004

-----
Process exited after 9.758 seconds with return value 0
Press any key to continue . . . |
```

B) Create A C++ Class Employee With Data Members Emp_Id, Emp Name, Mobile No, Salary. Write Necessary Member Functions For The Following:

- i. Accept details of n employees**
 - ii. Display employee details in descending order of their salary.**
 - iii. Display details of a particular employee.**
- (Use Array of object and Use appropriate manipulators)**

```
#include <iostream>
#include <iomanip>
#include <algorithm>
class Employee {
private:
    int Emp_Id;
    std::string Emp_Name;
    std::string Mobile_No;
    double Salary;

public:
    Employee() {}
    Employee(int id, std::string name, std::string mobile, double salary) :
        Emp_Id(id), Emp_Name(name), Mobile_No(mobile),
        Salary(salary) {}
    static void acceptDetails(Employee employees[], int n) {
        for (int i = 0; i < n; ++i) {
            int id;
            std::string name, mobile;
            double salary;
            std::cout << "Enter details for Employee " << i + 1 <<
            "\n";
            std::cout << "Employee ID: ";
            std::cin >> id;
            std::cout << "Employee Name: ";
            std::cin.ignore();
            std::getline(std::cin, name);
            std::cout << "Mobile No: ";
            std::cin >> mobile;
            std::cout << "Salary: ";
```

```

        std::cin >> salary;

        employees[i] = Employee(id, name, mobile, salary);
    }
}
static bool compareBySalary(const Employee &a, const Employee
&b) {
    return a.Salary > b.Salary;
}
static void displayDescendingOrder(Employee employees[], int n)
{
    std::sort(employees, employees + n, compareBySalary);
    std::cout << std::setw(10) << "Emp ID" << std::setw(20) <<
"Emp Name" << std::setw(15) << "Mobile No" << std::setw(10) <<
"Salary" << std::endl;
    for (int i = 0; i < n; ++i) {
        std::cout << std::setw(10) << employees[i].Emp_Id <<
std::setw(20) << employees[i].Emp_Name << std::setw(15) <<
employees[i].Mobile_No << std::setw(10) << employees[i].Salary
<< std::endl;
    }
}
static void displayEmployeeDetails(Employee employees[], int n,
int id) {
    bool found = false;
    for (int i = 0; i < n; ++i) {
        if (employees[i].Emp_Id == id) {
            found = true;
            std::cout << "Employee Details:\n";
            std::cout << "Employee ID: " << employees[i].Emp_Id
<< std::endl;
            std::cout << "Employee Name: " <<
employees[i].Emp_Name << std::endl;
            std::cout << "Mobile No: " << employees[i].Mobile_No
<< std::endl;
            std::cout << "Salary: " << employees[i].Salary <<
std::endl;
            break;
        }
    }
}

```

```

    }
    if (!found) {
        std::cout << "Employee with ID " << id << " not found.\n";
    }
}
};

int main() {
    int n;
    std::cout << "Enter the number of employees: ";
    std::cin >> n;
    Employee* employees = new Employee[n];
    Employee::acceptDetails(employees, n);
    std::cout << "\nEmployee Details in Descending Order of
Salary:\n";
    Employee::displayDescendingOrder(employees, n);
    int searchId;
    std::cout << "\nEnter Employee ID to search details: ";
    std::cin >> searchId;
    Employee::displayEmployeeDetails(employees, n, searchId);
    delete[] employees;
    return 0;
}

```

The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 2". The program prompts the user to enter the number of employees (2) and then details for each employee. The details for Employee 1 are: ID: 101, Name: Dear, Mobile No: 0000000000, Salary: 200000. The details for Employee 2 are: ID: 102, Name: Comrade, Mobile No: 0000000000, Salary: 180000. The program then displays the employee details in descending order of salary as a table:

Emp ID	Emp Name	Mobile No	Salary
101	Dear	0000000000	200000
102	Comrade	0000000000	180000

Next, the program prompts the user to enter an employee ID to search details (101). It then displays the details for Employee 1: ID: 101, Name: Dear, Mobile No: 0000000000, Salary: 200000. The program ends with a message: "Process exited after 107.8 seconds with return value 0. Press any key to continue . . . |".

Slip 29

A) Write a C++ program to create a class E_Bill with data members Cust_Name, Meter_ID, No_of_Units and Total_Charges. Write member functions to accept and display customer information by calculating charges. (Rules to calculate electricity board charges)

- For first 100 units Rs. 1 per unit
- For next 200 units Rs. 2 per unit
- Beyond 300 units Rs. 5 per unit

All users are charged a minimum of Rs.150. If the total charge is more than Rs.250.00 then an additional charge of 15% is added.

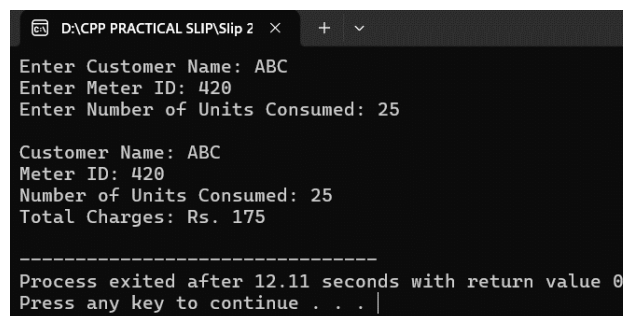
```
#include <iostream>
#include <string>
using namespace std;
class E_Bill {
private:
    string Cust_Name;
    int Meter_ID;
    int No_of_Units;
    double Total_Charges;
public:
    E_Bill() {
        Cust_Name = "";
        Meter_ID = 0;
        No_of_Units = 0;
        Total_Charges = 0.0;
    }
    void acceptInfo() {
        cout << "Enter Customer Name: ";
        getline(cin, Cust_Name);
        cout << "Enter Meter ID: ";
        cin >> Meter_ID;
        cout << "Enter Number of Units Consumed: ";
        cin >> No_of_Units;
        cin.ignore();
        calculateCharges();
    }
}
```

```

void calculateCharges() {
    double base_charge = 150.0;
    if (No_of_Units <= 100)
        Total_Charges = base_charge + (No_of_Units * 1.0);
    else if (No_of_Units <= 300)
        Total_Charges = base_charge + (100 * 1.0) + ((No_of_Units
- 100) * 2.0);
    else
        Total_Charges = base_charge + (100 * 1.0) + (200 * 2.0) +
((No_of_Units - 300) * 5.0);
    if (Total_Charges > 250.0)
        Total_Charges += (0.15 * Total_Charges);
}
void displayInfo() {
    cout << "\nCustomer Name: " << Cust_Name << endl;
    cout << "Meter ID: " << Meter_ID << endl;
    cout << "Number of Units Consumed: " << No_of_Units <<
endl;
    cout << "Total Charges: Rs. " << Total_Charges << endl;
}
};
int main() {
    E_Bill bill;
    bill.acceptInfo();
    bill.displayInfo();

    return 0;
}

```



The screenshot shows a Windows command prompt window titled "D:\C++ PRACTICAL SLIP\Slip 2". The user has entered the following inputs: "ABC" for Customer Name, "420" for Meter ID, and "25" for Number of Units Consumed. The program has calculated and displayed the following output: "Customer Name: ABC", "Meter ID: 420", "Number of Units Consumed: 25", and "Total Charges: Rs. 175". At the bottom, it shows "Process exited after 12.11 seconds with return value 0" and "Press any key to continue . . .".

```

D:\C++ PRACTICAL SLIP\Slip 2 >
Enter Customer Name: ABC
Enter Meter ID: 420
Enter Number of Units Consumed: 25

Customer Name: ABC
Meter ID: 420
Number of Units Consumed: 25
Total Charges: Rs. 175

-----
Process exited after 12.11 seconds with return value 0
Press any key to continue . . .

```

B) Create a C++ class VisitingStaff with data members Name, No_of_Subjects, Name_of_Subjects[], Working_hours, Total_Salary. (Number of subjects varies for a Staff). Write a parameterized constructor to initialize the data members and create an array for Name_of_Subjects dynamically. Display Visiting Staff details by calculating salary. (Assume remuneration Rs. 300 per working hour)..

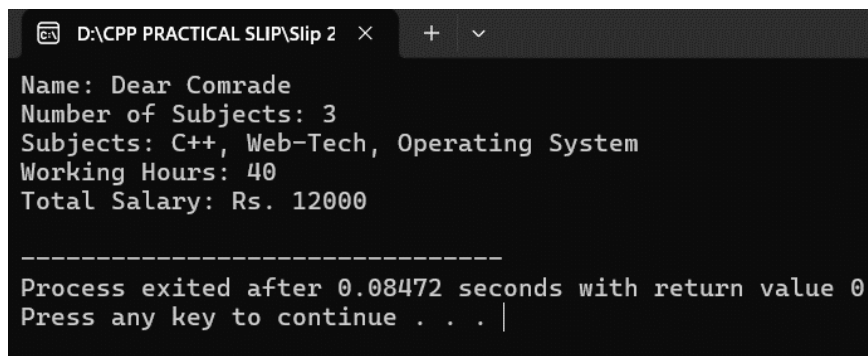
```
#include <iostream>
#include <string>
class VisitingStaff {
private:
    std::string Name;
    int No_of_Subjects;
    std::string *Name_of_Subjects;
    int Working_hours;
    double Total_Salary;
public:
    VisitingStaff(std::string name, int num_subjects, std::string
subjects[], int hours)
        :Name(name),No_of_Subjects(num_subjects),
Working_hours(hours) {
        Name_of_Subjects = new std::string[num_subjects];
        for (int i = 0; i < num_subjects; ++i) {
            Name_of_Subjects[i] = subjects[i];
        }
        Total_Salary = hours * 300;
    }
    ~VisitingStaff() {
        delete[] Name_of_Subjects;
    }
    void displayDetails() {
        std::cout << "Name: " << Name << std::endl;
        std::cout << "Number of Subjects: " << No_of_Subjects <<
std::endl;
        std::cout << "Subjects: ";
        for (int i = 0; i < No_of_Subjects; ++i) {
            std::cout << Name_of_Subjects[i];
            if (i != No_of_Subjects - 1)
```



```

        std::cout << ", ";
    }
    std::cout << std::endl;
    std::cout << "Working Hours: " << Working_hours <<
std::endl;
    std::cout << "Total Salary: Rs. " << Total_Salary << std::endl;
    }
};
int main() {
    std::string subjects[] = {"C++", "Web-Tech", "Operating
System"};
    VisitingStaff staff("Dear Comrade", 3, subjects, 40);
    staff.displayDetails();
    return 0;
}

```



The screenshot shows a terminal window titled "D:\CPP PRACTICAL SLIP\Slip 2". The output of the program is as follows:

```

Name: Dear Comrade
Number of Subjects: 3
Subjects: C++, Web-Tech, Operating System
Working Hours: 40
Total Salary: Rs. 12000

-----
Process exited after 0.08472 seconds with return value 0
Press any key to continue . . . |

```

Slip 30

A) Write C++ Program To Create Two Classes Integer Array And Float Array With An Array As A Data Member. Write Necessary Member Functions To Accept And Display Array Elements Of Both The Classes. Find And Display Average Of Both The Array. (Use Friend Function)...

```
#include <iostream>
class IntegerArray;
class FloatArray;
class IntegerArray {
private:
    int size;
    int *arr;
public:
    IntegerArray(int s) : size(s) {
        arr = new int[size];
    }
    void acceptArray() {
        std::cout << "Enter " << size << " integer elements:\n";
        for (int i = 0; i < size; ++i) {
            std::cout << "Element " << i + 1 << ": ";
            std::cin >> arr[i];
        }
    }
    void displayArray() {
        std::cout << "Integer Array: ";
        for (int i = 0; i < size; ++i) {
            std::cout << arr[i] << " ";
        }
        std::cout << std::endl;
    }
    friend float findAverage(const IntegerArray &intArr, const
FloatArray &floatArr);
};
class FloatArray {
private:
    int size;
    float *arr;
```

```

public:
    FloatArray(int s) : size(s) {
        arr = new float[size];
    }
    void acceptArray() {
        std::cout << "Enter " << size << " float elements:\n";
        for (int i = 0; i < size; ++i) {
            std::cout << "Element " << i + 1 << ": ";
            std::cin >> arr[i];
        }
    }
    void displayArray() {
        std::cout << "Float Array: ";
        for (int i = 0; i < size; ++i) {
            std::cout << arr[i] << " ";
        }
        std::cout << std::endl;
    }
    friend float findAverage(const IntegerArray &intArr, const
FloatArray &floatArr);
};
float findAverage(const IntegerArray &intArr, const FloatArray
&floatArr) {
    float intSum = 0, floatSum = 0;
    for (int i = 0; i < intArr.size; ++i) {
        intSum += intArr.arr[i];
    }
    for (int i = 0; i < floatArr.size; ++i) {
        floatSum += floatArr.arr[i];
    }
    return (intSum / intArr.size + floatSum / floatArr.size) / 2;
}
int main() {
    int intSize, floatSize;
    std::cout << "Enter size of integer array: ";
    std::cin >> intSize;
    IntegerArray intArr(intSize);
    intArr.acceptArray();
    intArr.displayArray();
}

```

```

std::cout << "Enter size of float array: ";
std::cin >> floatSize;
FloatArray floatArr(floatSize);
floatArr.acceptArray();
floatArr.displayArray();
std::cout << "Average of both arrays: " << findAverage(intArr,
floatArr) << std::endl;
return 0;
}

```

```

D:\CPP PRACTICAL SLIP\Slip 3
Enter size of integer array: 4
Enter 4 integer elements:
Element 1: 12
Element 2: 11
Element 3: 29
Element 4: 20
Integer Array: 12 11 29 20
Enter size of float array:
4
Enter 4 float elements:
Element 1: 12.1
Element 2: 11.2
Element 3: 29.3
Element 4: 20.4
Float Array: 12.1 11.2 29.3 20.4
Average of both arrays: 18.125

-----
Process exited after 39.72 seconds with return value 0
Press any key to continue . . . |

```

B) Create A C++ Class Marksheet With Data Members Seat_No, Student_Name,Class,Subject_Name[],Int_Marks[],Ext_Marks[], Total[], Grand_Total, Percentage, Grade. Write Member Function To Accept Student Information For 5 Subjects. Calculate Total, Grand_Total, Percentage, Grade And Use Setw(), Setprecision()And Setfill()To Display Marksheet...

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class Marksheet {
private:
    string seatNo;
    string studentName;
    string className;
    string subjectName[6];
    int intMarks[6];
    int extMarks[6];
    int totalMarks[6];
    int grandTotal;
    float percentage;
    char grade;
    void calculateTotalMarks() {
        for (int i = 0; i < 6; ++i) {
            totalMarks[i] = intMarks[i] + extMarks[i];
        }
    }
    void calculateGrandTotal() {
        grandTotal = 0;
        for (int i = 0; i < 6; ++i) {
            grandTotal += totalMarks[i];
        }
    }
    void calculatePercentage() {
        percentage = static_cast<float>(grandTotal) / 600 * 100;
    }
    void calculateGrade() {
        if (percentage >= 90)
```

```

        grade = 'A';
    else if (percentage >= 80)
        grade = 'B';
    else if (percentage >= 70)
        grade = 'C';
    else if (percentage >= 60)
        grade = 'D';
    else
        grade = 'F';
}
public:
void acceptInfo() {
    cout << "Enter Seat No: ";
    cin >> seatNo;
    cout << "Enter Student Name: ";
    cin.ignore();
    getline(cin, studentName);
    cout << "Enter Class: ";
    getline(cin, className);
    for (int i = 0; i < 6; ++i) {
        cout << "Enter Subject Name for Subject " << i + 1 << ": ";
        getline(cin, subjectName[i]);
        cout << "Enter Internal Marks for Subject " << i + 1 << ": ";
        cin >> intMarks[i];
        cout << "Enter External Marks for Subject " << i + 1 << ": ";
        cin >> extMarks[i];
        cin.ignore();    } }
void calculate() {
    calculateTotalMarks();
    calculateGrandTotal();
    calculatePercentage();
    calculateGrade();
}
void display() {
    cout << "\n\n***** Marksheet *****\n\n";
    cout << "Seat No: " << seatNo << endl;
    cout << "Student Name: " << studentName << endl;
    cout << "Class: " << className << endl;

```

```

cout << setw(20) << setfill(' ') << left << "Subject";
cout << setw(15) << right << "Internal";
cout << setw(15) << right << "External";
cout << setw(15) << right << "Total\n";
cout << setfill('-') << setw(45) << "-" << endl;
cout << setfill(' ');
for (int i = 0; i < 6; ++i) {
    cout << setw(20) << left << subjectName[i];
    cout << setw(15) << right << intMarks[i];
    cout << setw(15) << right << extMarks[i];
    cout << setw(15) << right << totalMarks[i] << endl;
}
cout << setfill('-') << setw(45) << "-" << endl;
cout << "Grand Total: " << grandTotal << endl;
cout << "Percentage: " << fixed << setprecision(2) <<
percentage << "%" << endl;
cout << "Grade: " << grade << endl;
}
};
int main() {
    Marksheet marksheet;
    marksheet.acceptInfo();
    marksheet.calculate();
    marksheet.display();
    return 0;}

```

The screenshot shows a Windows command prompt window titled "D:\CPP PRACTICAL SLIP\Slip 3". The program prompts for the following inputs:

- Enter Seat No: 00000
- Enter Student Name: RHTDM
- Enter Class: 12th
- Enter Subject Name for Subject 1: English
- Enter Internal Marks for Subject 1: 25
- Enter External Marks for Subject 1: 65
- Enter Subject Name for Subject 2: Hindi
- Enter Internal Marks for Subject 2: 28
- Enter External Marks for Subject 2: 70
- Enter Subject Name for Subject 3: Mathematics
- Enter Internal Marks for Subject 3: 29
- Enter External Marks for Subject 3: 69
- Enter Subject Name for Subject 4: Chemistry
- Enter Internal Marks for Subject 4: 28
- Enter External Marks for Subject 4: 70
- Enter Subject Name for Subject 5: Biology
- Enter Internal Marks for Subject 5: 30
- Enter External Marks for Subject 5: 70
- Enter Subject Name for Subject 6: Physics
- Enter Internal Marks for Subject 6: 25
- Enter External Marks for Subject 6: 60

The output displays a formatted marksheet table:

```

***** Marksheet *****

Seat No: 00000
Student Name: RHTDM
Class: 12th

```

Subject	Internal	External	Total
English	25	65	90
Hindi	28	70	98
Mathematics	29	69	98
Chemistry	28	70	98
Biology	30	70	100
Physics	25	60	85

```

Grand Total: 569
Percentage: 94.83%
Grade: A

Process exited after 106.4 seconds with return value 0
Press any key to continue . . .

```