Question 1:

Write a program to print the ASCII value of a character

Input Format:

- Input consist of a character

Output Format:

- Output prints the ASCII value of the character

Title for Question 1: ASCII value

Solution:

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

// Define a class called ASCIIConverter
class ASCIIConverter {
private:
    char inputChar;  // Private member variable to store the input charac

public:
    // Member function to get the character from the user
    void getChar() {
        //cout << "Enter a character: ";
        cin >> inputChar;
    }

    // Member function to display the ASCII value of the character
    void displayASCIIValue() {
        cout << "The ASCII value of '" << inputChar << "' is " << int(inp
    }
};

int main() {
    // Create an object of ASCIIConverter class
    ASCIIConverter obj;

    // Call the getChar() method to accept input
    obj.getChar();

    // Call the displayASCIIValue() method to display the ASCII value
    obj.displayASCIIValue();

    return 0;
```

```
}
```

TestCases:

S.No	Inputs	Outputs
1	a	The ASCII value of 'a' is 97
2	M	The ASCII value of 'M' is 77
3	i	The ASCII value of 'i' is 105
4	#	The ASCII value of '#' is 35
5	H	The ASCII value of 'H' is 72
6	o	The ASCII value of 'o' is 111

White List:

Black List:

Question 2:

You are tasked with developing a library fines calculator system . The library has a policy for calculating fines on overdue books. The policy is as follows:

- For the first 7 days overdue: No fine
- For the next 7 days (8th to 14th day): Rs. 2 fine per day
- For the subsequent days (15th day and beyond): Rs. 5 fine per day

Design a program that takes inputs regarding the number of days a book is overdue and calculates the fine amount. Use class and resolution operator to implement the solution.

Input Format:

- Enter the number of days the book is overdue.

Output Format:

- If no fine is applicable, print "No Fine".
- If the fine is applicable, print the calculated fine amount in Rupees.

Title for Question 2: Library fine calculator

Solution:

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

```
class LibraryFines {
private:
    int daysOverdue;

public:
    // Constructor to initialize daysOverdue
    LibraryFines(int days);

    // Member function to calculate and display the fine
    void calculateFine();
};

// Define the constructor using the resolution operator
LibraryFines::LibraryFines(int days) : daysOverdue(days) {}

// Define the member function using the resolution operator
void LibraryFines::calculateFine() {
    int fine = 0;
    if (daysOverdue > 7 && daysOverdue <= 14) {
        fine = 2 * (daysOverdue - 7);
    } else if (daysOverdue > 14) {
        fine = 2 * 7 + 5 * (daysOverdue - 14);
    }

    cout << "Fine: Rs. " << fine << endl;
}

int main() {
    int days;

    //cout << "Enter the number of days overdue: ";
    cin >> days;

    LibraryFines finesCalculator(days);
    finesCalculator.calculateFine();

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	12	Fine: Rs. 10
2	5	Fine: Rs. 0
3	66	Fine: Rs. 274
4	8	Fine: Rs. 2
5	23	Fine: Rs. 59
6	100	Fine: Rs. 444

White List:

Black List:

Question 3:

Create a class Outer with a private member variable x. Define a nested class Inner with a private member variable y. Implement methods to set and display both x and y. Write a program to create an Outer object and access the nested Inner class using the resolution operator.

?????Input Format:

- The program expects two integer values as input:
- Enter the value for x, a private member variable of the Outer class.
- Enter the value for y, a private member variable of the nested Inner class.

Output Format:

- The program displays the following output:
- The value of x using the method displayOuter of the Outer class.
- The value of y using the method displayInner of the nested Inner class.

Title for Question 3: Inner - Outer class

Solution:

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

class Outer {
private:
    int x;

public:
    Outer(int val) : x(val) {}

    class Inner {
    private:
        int y;

    public:
        Inner(int val) : y(val) {}

        void displayInner() {
            cout << "Inner y: " << y << endl;
        }
    };

    void setX(int val) {
        x = val;
    }

    void displayOuter() {
        cout << "Outer x: " << x << endl;
    }
```

```
};

int main() {
    int x_val, y_val;

    //cout << "Enter x: ";
    cin >> x_val;

   // cout << "Enter y: ";
    cin >> y_val;

    Outer outer(x_val);
    Outer::Inner inner(y_val);

    outer.displayOuter();
    inner.displayInner();

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	10 20	Outer x: 10 Inner y: 20
2	9 5	Outer x: 9 Inner y: 5
3	123 54	Outer x: 123 Inner y: 54
4	9876543 12345	Outer x: 9876543 Inner y: 12345
5	567890 000000	Outer x: 567890 Inner y: 0
6	846534 1936855	Outer x: 846534 Inner y: 1936855

White List:

Black List:

Question 4:

Implement a class named Employee with private member variables for 'name' (string), 'employee ID' (integer), and 'salary' (double). Create member functions to calculate and set the salary based on an employee's performance rating.

Input Format:

- Input employee name: [User enters the employee's name as a string]
- Input employee ID: [User enters the employee's ID as an integer]
- Input employee salary: [User enters the employee's salary as a double]
- Input performance rating (0.0-1.4): [User enters the performance rating as a double]

Output Format:

- Initial salary: $[Initial Salary]

- Input performance rating (0.0-1.4): [User enters the performance rating as a double]
- Invalid performance rating. Salary remains unchanged. (if performance rating is invalid)
- Updated salary: $[Updated Salary]
- The program prompts the user for input, displays the entered values, and provides output messages about the initial and updated salary based on performance, including handling invalid performance ratings.

Title for Question 4: Employee salary update

Solution:

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

class Employee {
protected:
    std::string name;
    int employeeId;
    double salary;

public:
    // Constructor
    Employee(const std::string& empName, int empId, double empSalary)
        : name(empName), employeeId(empId), salary(empSalary) {}

    // Member function to calculate salary based on performance
    void calculateSalary(double performanceRating) {
        if (performanceRating >= 0.0 && performanceRating <= 1.4) {
            salary *= performanceRating;
        } else {
            std::cout << "Invalid performance rating. Salary remains unch
        }
    }

    // Member functions to get employee details
    std::string getName() const {
        return name;
    }

    int getEmployeeId() const {
        return employeeId;
    }

    double getSalary() const {
        return salary;
    }
};

int main() {
    // Create an employee object
    std::string empName;
    int empId;
    double empSalary;
    double performanceRating = 0.0;

 //   std::cout << "Input employee name: ";
```

```
    std::getline(std::cin, empName);

  //  std::cout << "Input employee ID: ";
    std::cin >> empId;

 //   std::cout << "Input employee salary: ";
    std::cin >> empSalary;

    Employee employee(empName, empId, empSalary);

    // Display initial salary
    std::cout << "Initial salary: " << employee.getSalary() << std::endl;

    // Clear std::cin and ignore remaining characters
    std::cin.clear();
    std::cin.ignore(10000, '\n');

   // std::cout << "Input performance rating (0.0-1.4): ";
    std::cin >> performanceRating;

    employee.calculateSalary(performanceRating);

    // Display updated salary
    std::cout << "Updated salary: " << employee.getSalary() << std::endl;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	John Doe 101 50000 1.2	Initial salary: 50000 Updated salary: 60000
2	Jane Smith 102 60000 -0.5	Initial salary: 60000 Invalid performance rating. Salary remains unchanged. Updated salary: 60000
3	Robert Johnson 103 70000 1.6	Initial salary: 70000 Invalid performance rating. Salary remains unchanged. Updated salary: 70000
4	Mary Brown 104 55000 1.0	Initial salary: 55000 Updated salary: 55000
5	Michael Lee 105 60000.0 1.5	Initial salary: 60000 Invalid performance rating. Salary remains unchanged. Updated salary: 60000
6	Lisa Green 106 65000.0 0.0	Initial salary: 65000 Updated salary: 0

White List:

Black List:

Question 5:

Build a code to handle multiple students and dynamically accept user input for an arbitrary number of students. Each student's information should be stored in an array or a container of your choice. After collecting the information for all the students, display their details, including name, class, roll

number, marks, and grade. Ensure that your code can handle an unknown number of students until the user decides to stop entering data.

Input Format:

- For each student, the program should accept the following input:
- Student's name (a string with spaces).
- Student's class (a string with spaces).
- Roll number (an integer).
- Marks (a double).

Output Format:

- After collecting information for all students, display each student's details, including name, class, roll number, marks, and grade, in the following format:
- Name: [Student 1 Name]
- Class: [Student 1 Class]
- Roll Number: [Student 1 Roll Number]
- Marks: [Student 1 Marks]
- Grade: [Student 1 Grade]

Title for Question 5: student grade system

Solution:

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


class Student {
private:
    std::string name;
    std::string studentClass;
    int rollNumber;
    double marks;
public:
    // Constructor to initialize the Student object
    Student(std::string n, std::string cls, int roll, double m) : name(n)


    // Member function to calculate the grade based on marks
    char calculateGrade() {
        if (marks >= 90)
            return 'A';
        else if (marks >= 80)
            return 'B';
```

```
        else if (marks >= 70)
            return 'C';
        else if (marks >= 60)
            return 'D';
        else
            return 'F';
    }


    // Member function to display student information
    void displayInfo() {
        std::cout << "Name: " << name << std::endl;
        std::cout << "Class: " << studentClass << std::endl;
        std::cout << "Roll Number: " << rollNumber << std::endl;
        std::cout << "Marks: " << marks << std::endl;
        std::cout << "Grade: " << calculateGrade() << std::endl;
    }
};


int main() {
    std::string name, studentClass;
    int rollNumber;
    double marks;


    // Input student information


    std::getline(std::cin, name);



    std::getline(std::cin, studentClass);


    std::cin >> rollNumber;


    std::cin >> marks;


    // Create a Student object with user input
    Student student(name, studentClass, rollNumber, marks);


    // Display student information
    student.displayInfo();


    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	John Doe 10A 101 95.5	Name: John Doe Class: 10A Roll Number: 101 Marks: 95.5 Grade: A

S.No	Inputs	Outputs
2	Alice Smith 11B 202 78.5	Name: Alice Smith Class: 11B Roll Number: 202 Marks: 78.5 Grade: C
3	Bob Johnson 9C 303 85.0	Name: Bob Johnson Class: 9C Roll Number: 303 Marks: 85 Grade: B
4	Mary David 12A 404 67.5	Name: Mary David Class: 12A Roll Number: 404 Marks: 67.5 Grade: D
5	Emma Lee 8B 505 55.0	Name: Emma Lee Class: 8B Roll Number: 505 Marks: 55 Grade: F
6	Michael Brown 10A 606 72.5	Name: Michael Brown Class: 10A Roll Number: 606 Marks: 72.5 Grade: C

White List:

Black List:

Question 6:

Build a code to allow the user to input information for multiple persons and display their details. The program should continue to prompt the user for information until they choose to stop. After each person's information is entered, the program should ask the user if they want to input details for another person. The input and output format should be as follows:

Input Format:

- For each person:
- Enter name (a string with no spaces).
- Enter age (an integer).
- Enter country (a string with no spaces).

Output Format:

- For each person:
- Display their information as follows:
- Person Info:
- Name: [Name]
- Age: [Age]
- Country: [Country]

Title for Question 6: Person info

Solution:

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

class Person {
private:
    std::string name;
    int age;
    std::string country;

public:
    void setName(std::string n) {
        name = n;
    }

    void setAge(int a) {
        age = a;
    }

    void setCountry(std::string c) {
        country = c;
    }

    std::string getName() {
        return name;
    }

    int getAge() {
        return age;
    }

    std::string getCountry() {
        return country;
    }
};

int main() {
    Person p;

    std::string name, country;
    int age;

  //  std::cout << "Enter name: ";
    std::getline(std::cin, name);
    p.setName(name);

  //  std::cout << "Enter age: ";
    std::cin >> age;
    p.setAge(age);

    std::cin.ignore();
    //std::cout << "Enter country: ";
    std::getline(std::cin, country);
    p.setCountry(country);

    std::cout << "Person Info:" << std::endl;
    std::cout << "Name: " << p.getName() << std::endl;
    std::cout << "Age: " << p.getAge() << std::endl;
    std::cout << "Country: " << p.getCountry() << std::endl;

    return 0;
```

```
}
```

TestCases:

S.No	Inputs	Outputs
1	John 30 USA	Person Info: Name: John Age: 30 Country: USA
2	John 0 USA	Person Info: Name: John Age: 0 Country: USA
3	John 25 United	Person Info: Name: John Age: 25 Country: United
4	John 25 USA	Person Info: Name: John Age: 25 Country: USA
5	John 25 United	Person Info: Name: John Age: 25 Country: United
6	John -30 USA	Person Info: Name: John Age: -30 Country: USA

White List:

Black List:
