SAVEETHA SCHOOL OF ENGINEERING

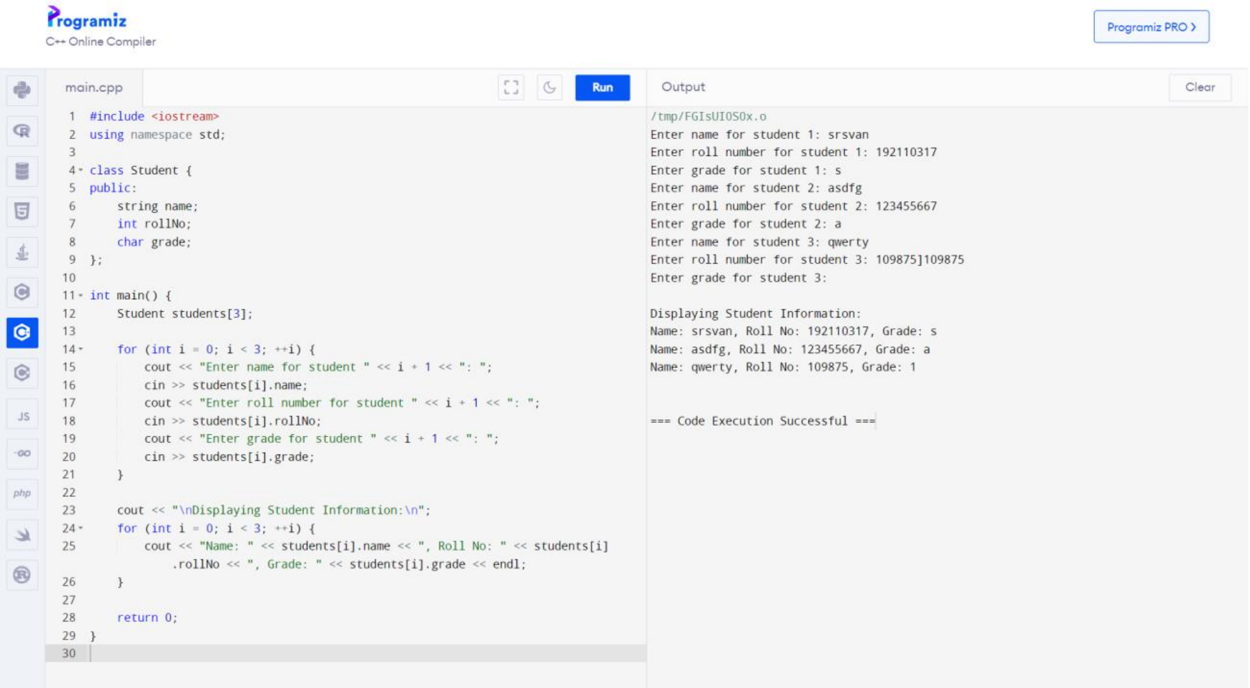
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

Engineer to Excel

SUB CODE & NAME : DSA01/ Object Oriented Programming with C++

LAB DAY 6/21-03-2024

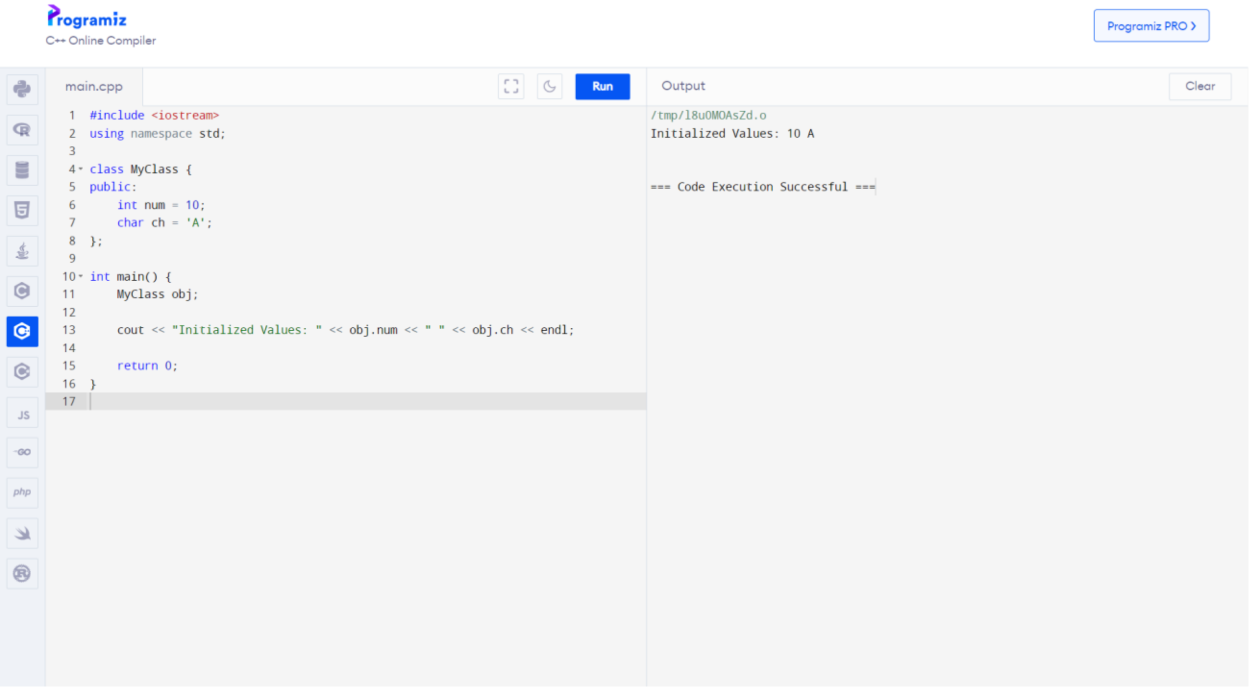
1. Write a C++ Program to display Names, Roll No., and grades of 3 students who have appeared in the examination. Declare the class of name, Roll No. and grade. Create an array of class objects. Read and display the contents of the array.



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

class Student {
public:
    string name;
    int rollNo;
    char grade;
};

int main() {
    Student students[3];

    for (int i = 0; i < 3; ++i) {
        cout << "Enter name for student " << i + 1 << ": ";
        cin >> students[i].name;
        cout << "Enter roll number for student " << i + 1 << ": ";
        cin >> students[i].rollNo;
        cout << "Enter grade for student " << i + 1 << ": ";
        cin >> students[i].grade;
    }

    cout << "\nDisplaying Student Information:\n";
    for (int i = 0; i < 3; ++i) {
        cout << "Name: " << students[i].name << ", Roll No: " << students[i]
            .rollNo << ", Grade: " << students[i].grade << endl;
    }

    return 0;
}

```

```
/tmp/FGIsUIOSOx.o
Enter name for student 1: srsvan
Enter roll number for student 1: 192110317
Enter grade for student 1: s
Enter name for student 2: asdfg
Enter roll number for student 2: 123455667
Enter grade for student 2: a
Enter name for student 3: qwerty
Enter roll number for student 3: 109875]109875
Enter grade for student 3:

Displaying Student Information:
Name: srsvan, Roll No: 192110317, Grade: s
Name: asdfg, Roll No: 123455667, Grade: a
Name: qwerty, Roll No: 109875, Grade: 1


=== Code Execution Successful ===
```

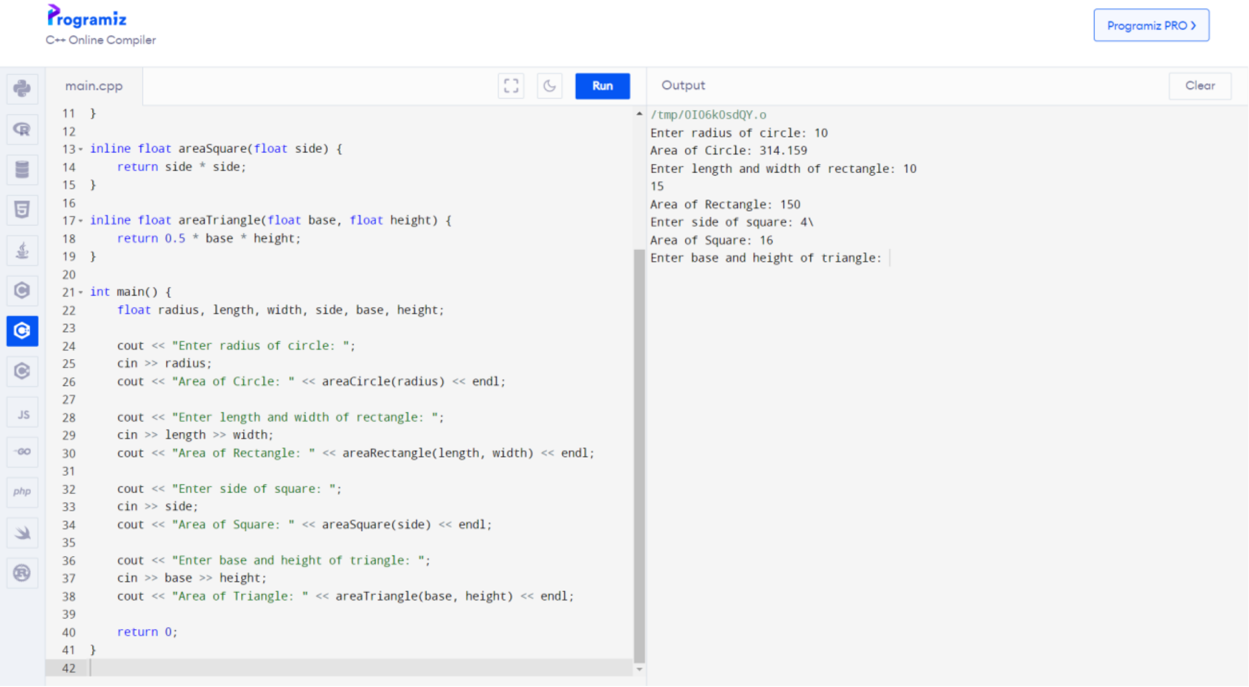
2. Write a C++ program to declare a class. Initialize and display the contents of the class member.



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

class MyClass {
public:
    int num = 10;
    char ch = 'A';
};

int main() {
    MyClass obj;

    cout << "Initialized Values: " << obj.num << " " << obj.ch << endl;

    return 0;
}
```

```
/tmp/l8u0MOAsZd.o
Initialized Values: 10 A


=== Code Execution Successful ===
```

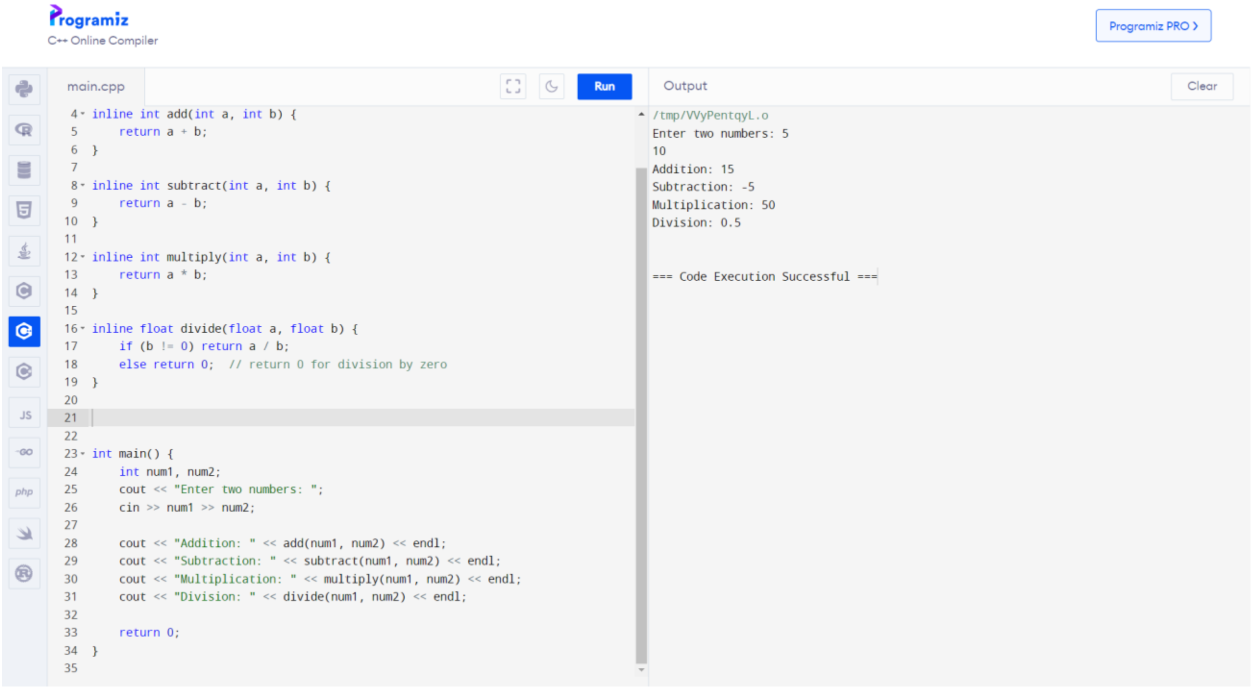
3. Write a program in C++ to calculate the area of circle, rectangle, square and triangle using inline function.



```cpp
}

inline float areaSquare(float side) {
    return side * side;
}

inline float areaTriangle(float base, float height) {
    return 0.5 * base * height;
}

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

    cout << "Enter radius of circle: ";
    cin >> radius;
    cout << "Area of Circle: " << areaCircle(radius) << endl;

    cout << "Enter length and width of rectangle: ";
    cin >> length >> width;
    cout << "Area of Rectangle: " << areaRectangle(length, width) << endl;

    cout << "Enter side of square: ";
    cin >> side;
    cout << "Area of Square: " << areaSquare(side) << endl;

    cout << "Enter base and height of triangle: ";
    cin >> base >> height;
    cout << "Area of Triangle: " << areaTriangle(base, height) << endl;

    return 0;
}
```

```
/tmp/0I06k0sdQY.o
Enter radius of circle: 10
Area of Circle: 314.159
Enter length and width of rectangle: 10
15
Area of Rectangle: 150
Enter side of square: 4\
Area of Square: 16
Enter base and height of triangle:
```

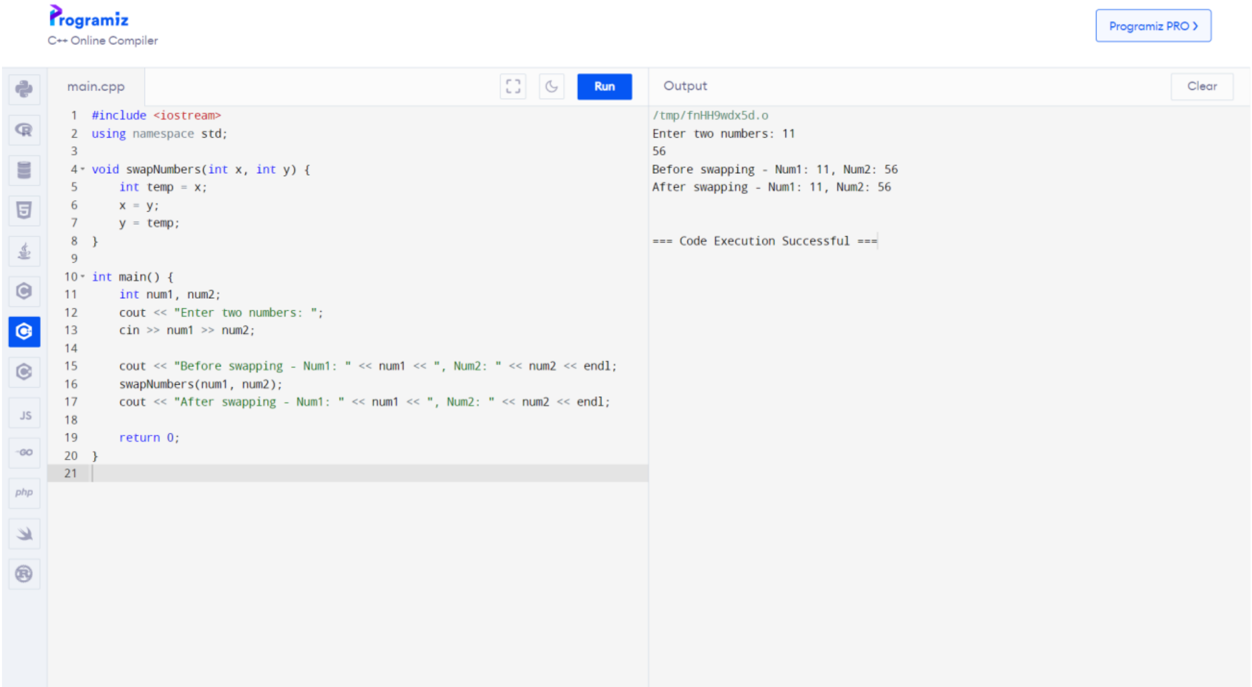
4. Write a C++ program to perform different arithmetic operations such as addition, subtraction, division, modulus and multiplication using inline function.



```cpp
inline int add(int a, int b) {
    return a + b;
}

inline int subtract(int a, int b) {
    return a - b;
}

inline int multiply(int a, int b) {
    return a * b;
}

inline float divide(float a, float b) {
    if (b != 0) return a / b;
    else return 0;  // return 0 for division by zero
}


int main() {
    int num1, num2;
    cout << "Enter two numbers: ";
    cin >> num1 >> num2;

    cout << "Addition: " << add(num1, num2) << endl;
    cout << "Subtraction: " << subtract(num1, num2) << endl;
    cout << "Multiplication: " << multiply(num1, num2) << endl;
    cout << "Division: " << divide(num1, num2) << endl;

    return 0;
}
```

```
/tmp/VVyPentqyL.o
Enter two numbers: 5
10
Addition: 15
Subtraction: -5
Multiplication: 50
Division: 0.5


=== Code Execution Successful ===
```

5. Write a C++ program to swap two number using call by value mechanism.



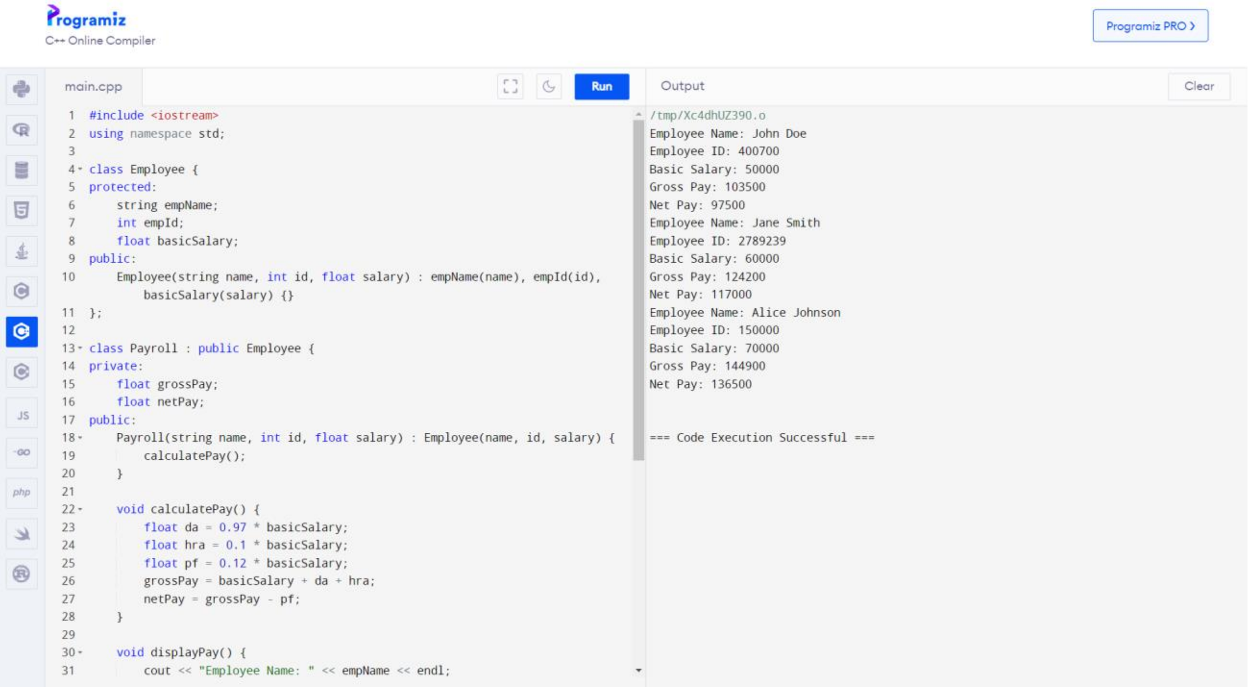
```cpp
#include <iostream>
using namespace std;

void swapNumbers(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}

int main() {
    int num1, num2;
    cout << "Enter two numbers: ";
    cin >> num1 >> num2;

    cout << "Before swapping - Num1: " << num1 << ", Num2: " << num2 << endl;
    swapNumbers(num1, num2);
    cout << "After swapping - Num1: " << num1 << ", Num2: " << num2 << endl;

    return 0;
}
```

```
/tmp/fnHH9wdx5d.o
Enter two numbers: 11
56
Before swapping - Num1: 11, Num2: 56
After swapping - Num1: 11, Num2: 56


=== Code Execution Successful ===
```

6. Write a c++ program to calculate the gross and net pay of employee from basic salary. Create employee which consists of employee name,emp_id, and basic salary as its data members . Use parameterized constructions in the derived class to initialize data mements of the base class and calculate gross and net pay of the employee in the derived class.

Test cases:

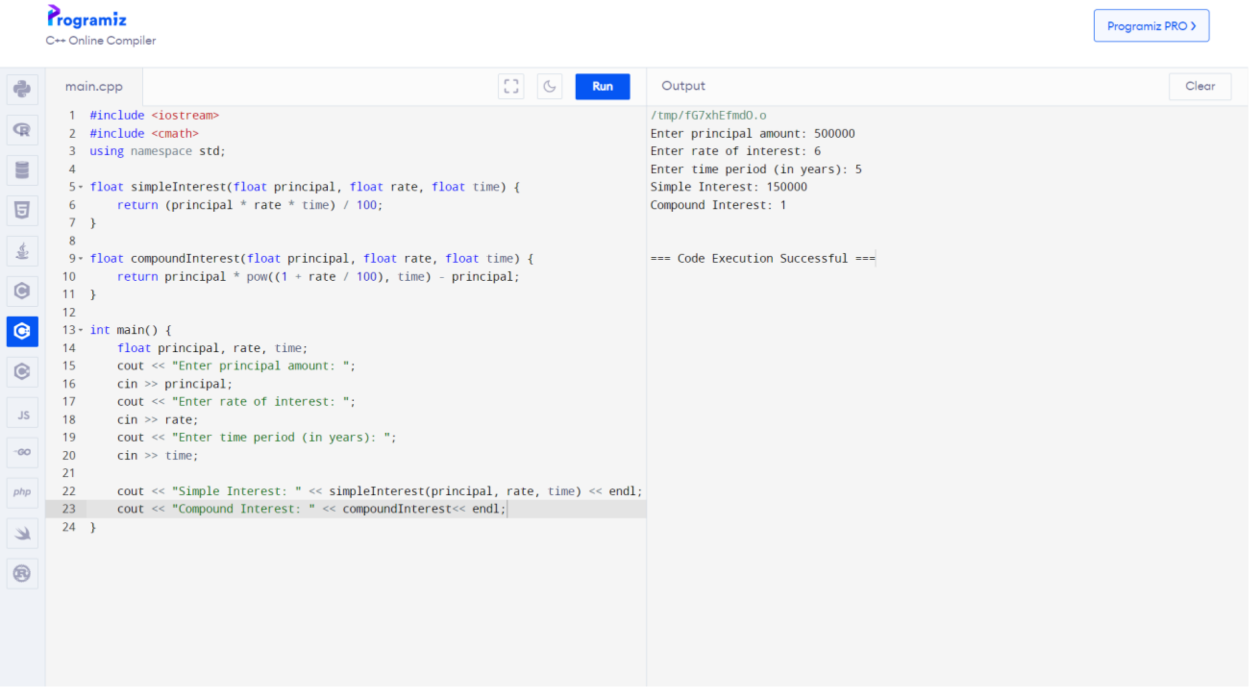
a. 400700
b. 2789239
c. 150000
d. 00000
e. -125486



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

class Employee {
protected:
    string empName;
    int empId;
    float basicSalary;
public:
    Employee(string name, int id, float salary) : empName(name), empId(id),
        basicSalary(salary) {}
};

class Payroll : public Employee {
private:
    float grossPay;
    float netPay;
public:
    Payroll(string name, int id, float salary) : Employee(name, id, salary) {
        calculatePay();
    }

    void calculatePay() {
        float da = 0.97 * basicSalary;
        float hra = 0.1 * basicSalary;
        float pf = 0.12 * basicSalary;
        grossPay = basicSalary + da + hra;
        netPay = grossPay - pf;
    }

    void displayPay() {
        cout << "Employee Name: " << empName << endl;
```

```
/tmp/Xc4dhUZ39O.o
Employee Name: John Doe
Employee ID: 400700
Basic Salary: 50000
Gross Pay: 103500
Net Pay: 97500
Employee Name: Jane Smith
Employee ID: 2789239
Basic Salary: 60000
Gross Pay: 124200
Net Pay: 117000
Employee Name: Alice Johnson
Employee ID: 150000
Basic Salary: 70000
Gross Pay: 144900
Net Pay: 136500


=== Code Execution Successful ===
```

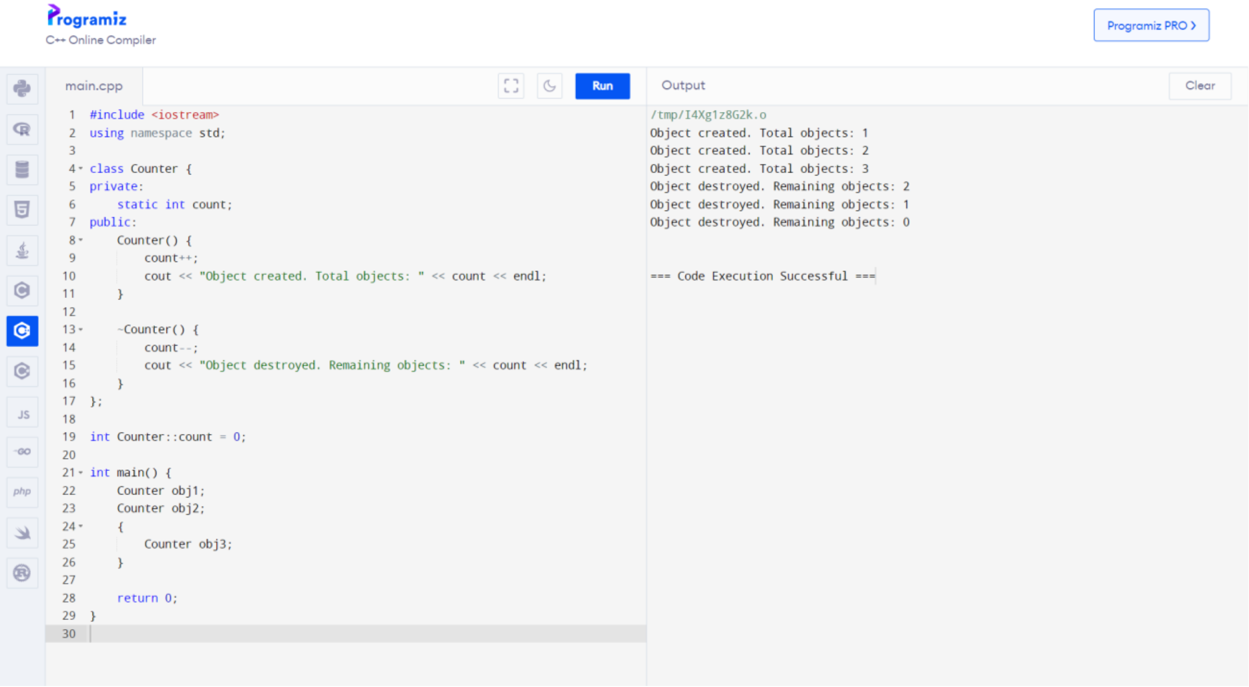
7. Write program in C++ to calculate simple interest and compound interest using default argument.



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

float simpleInterest(float principal, float rate, float time) {
    return (principal * rate * time) / 100;
}

float compoundInterest(float principal, float rate, float time) {
    return principal * pow((1 + rate / 100), time) - principal;
}

int main() {
    float principal, rate, time;
    cout << "Enter principal amount: ";
    cin >> principal;
    cout << "Enter rate of interest: ";
    cin >> rate;
    cout << "Enter time period (in years): ";
    cin >> time;

    cout << "Simple Interest: " << simpleInterest(principal, rate, time) << endl;
    cout << "Compound Interest: " << compoundInterest<< endl;
}
```

```
/tmp/fG7xhEfmdO.o
Enter principal amount: 500000
Enter rate of interest: 6
Enter time period (in years): 5
Simple Interest: 150000
Compound Interest: 1


=== Code Execution Successful ===
```

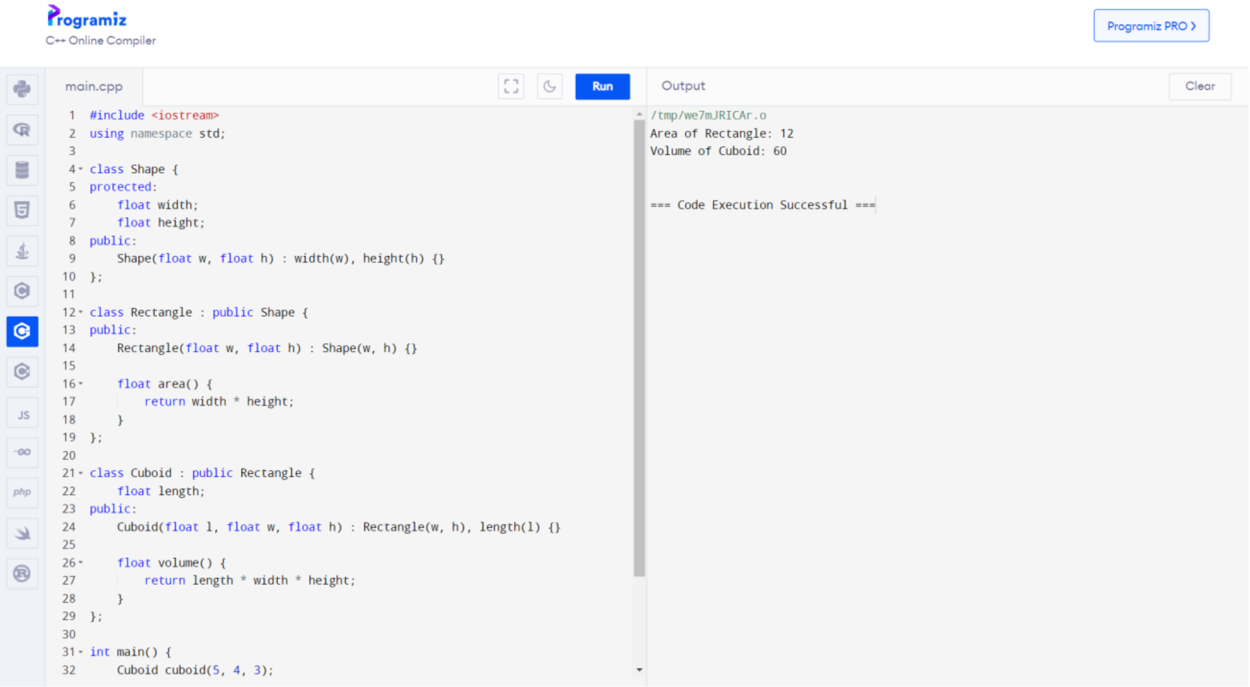
8. Create a Class for counting the number of objects created and destroyed within various block using constructor and destructor.



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

class Counter {
private:
    static int count;
public:
    Counter() {
        count++;
        cout << "Object created. Total objects: " << count << endl;
    }

    ~Counter() {
        count--;
        cout << "Object destroyed. Remaining objects: " << count << endl;
    }
};

int Counter::count = 0;

int main() {
    Counter obj1;
    Counter obj2;
    {
        Counter obj3;
    }

    return 0;
}

```

```
/tmp/I4Xg1z8G2k.o
Object created. Total objects: 1
Object created. Total objects: 2
Object created. Total objects: 3
Object destroyed. Remaining objects: 2
Object destroyed. Remaining objects: 1
Object destroyed. Remaining objects: 0


=== Code Execution Successful ===
```

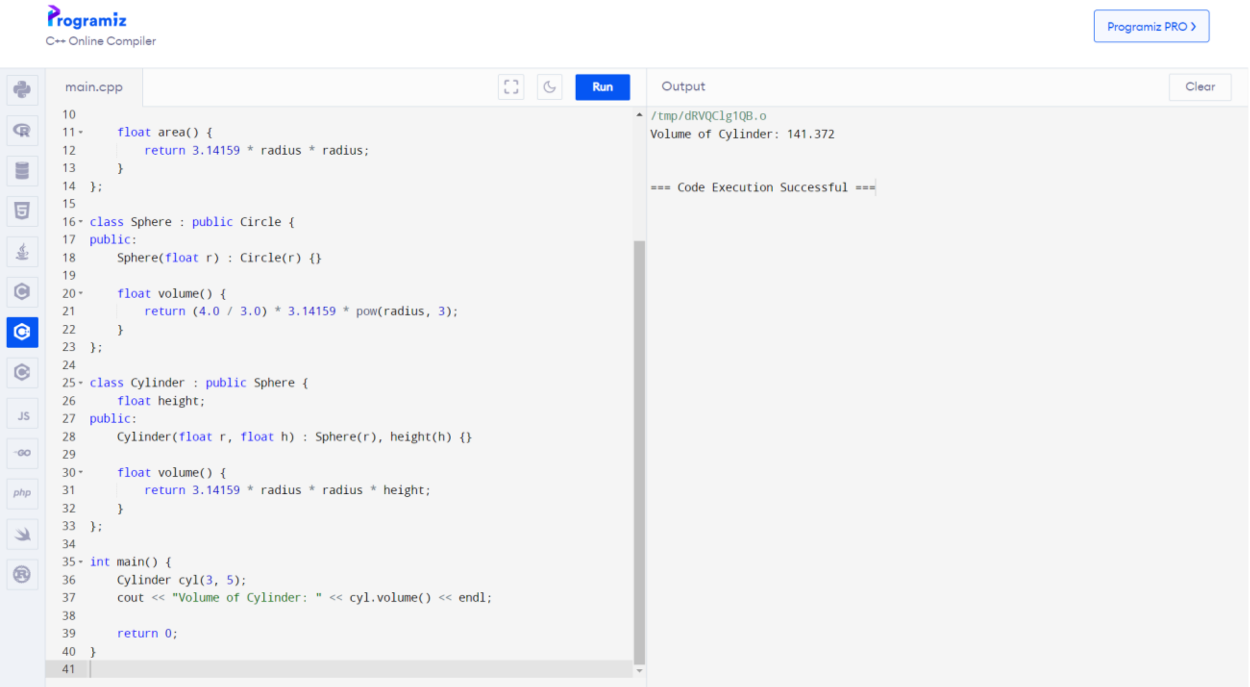
9. Write a C++ program to demonstrate the multiple inheritance by creating a class cuboid which extends class rectangle, class shape. It calculates area and volume.



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

class Shape {
protected:
    float width;
    float height;
public:
    Shape(float w, float h) : width(w), height(h) {}
};

class Rectangle : public Shape {
public:
    Rectangle(float w, float h) : Shape(w, h) {}

    float area() {
        return width * height;
    }
};

class Cuboid : public Rectangle {
    float length;
public:
    Cuboid(float l, float w, float h) : Rectangle(w, h), length(l) {}

    float volume() {
        return length * width * height;
    }
};

int main() {
    Cuboid cuboid(5, 4, 3);
```

```
/tmp/we7mJRICAr.o
Area of Rectangle: 12
Volume of Cuboid: 60


=== Code Execution Successful ===
```

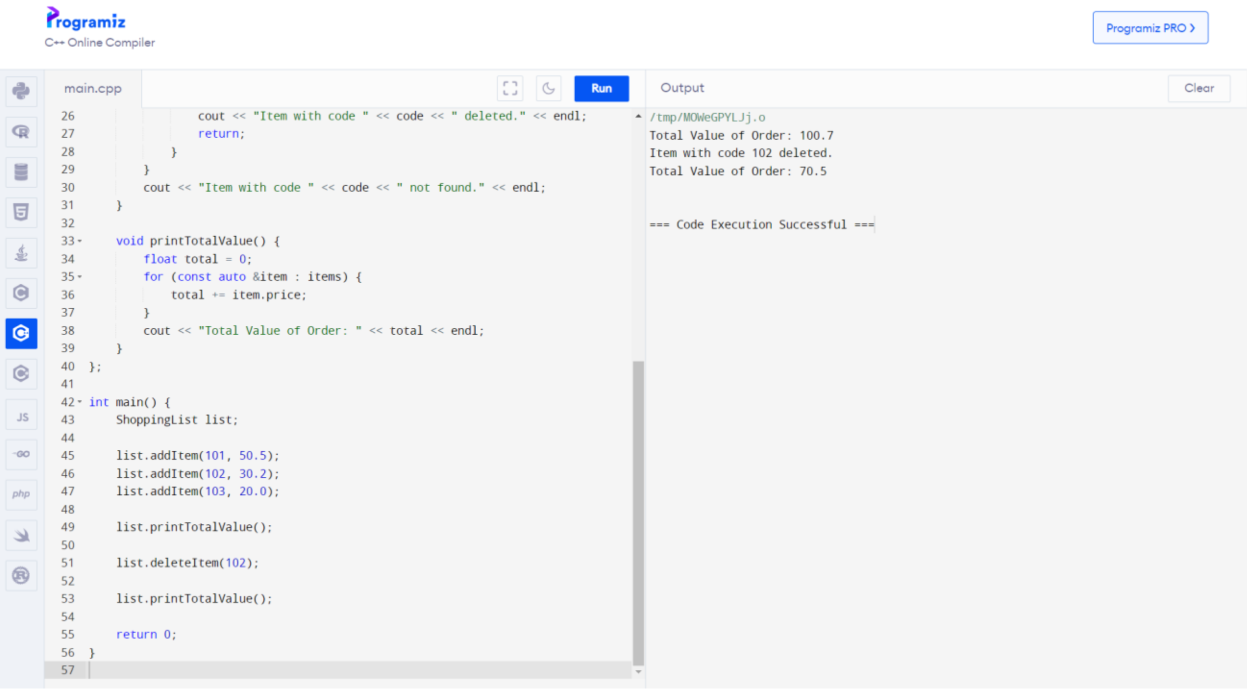
10. Create a class circle with data member radius; provide member function to calculate area. Derive a class sphere from class circle; provide member function to calculate volume.Derive class cylinder from class sphere with additional data member for height and member function to calculate volume.



```cpp

    float area() {
        return 3.14159 * radius * radius;
    }
};

class Sphere : public Circle {
public:
    Sphere(float r) : Circle(r) {}

    float volume() {
        return (4.0 / 3.0) * 3.14159 * pow(radius, 3);
    }
};

class Cylinder : public Sphere {
    float height;
public:
    Cylinder(float r, float h) : Sphere(r), height(h) {}

    float volume() {
        return 3.14159 * radius * radius * height;
    }
};

int main() {
    Cylinder cyl(3, 5);
    cout << "Volume of Cylinder: " << cyl.volume() << endl;

    return 0;
}

```

```
/tmp/dRVQClg1QB.o
Volume of Cylinder: 141.372

=== Code Execution Successful ===
```

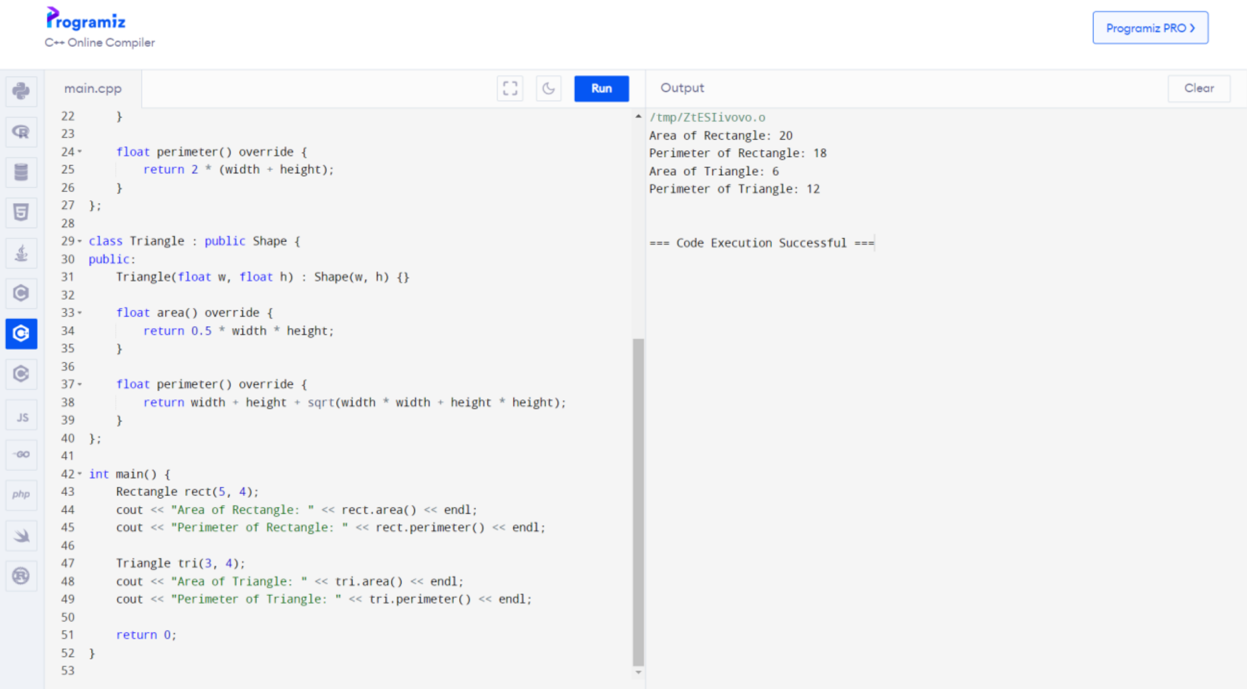
11. Write a Program using class to process Shopping List for a Departmental Store. The list include details such as the Code No and Price of each item and perform the operations like Adding, Deleting Items to the list and Printing the Total value of a Order.



```cpp
                cout << "Item with code " << code << " deleted." << endl;
                return;
            }
        }
        cout << "Item with code " << code << " not found." << endl;
    }

    void printTotalValue() {
        float total = 0;
        for (const auto &item : items) {
            total += item.price;
        }
        cout << "Total Value of Order: " << total << endl;
    }
};

int main() {
    ShoppingList list;

    list.addItem(101, 50.5);
    list.addItem(102, 30.2);
    list.addItem(103, 20.0);

    list.printTotalValue();

    list.deleteItem(102);

    list.printTotalValue();

    return 0;
}

```

```
/tmp/MOWeGPYLJj.o
Total Value of Order: 100.7
Item with code 102 deleted.
Total Value of Order: 70.5


=== Code Execution Successful ===
```

12. Create a base class called Shape with data members for height and width. Derive two classes Rectangle and Triangle from the base class. Write member functions to calculate the area and perimeter of each class.

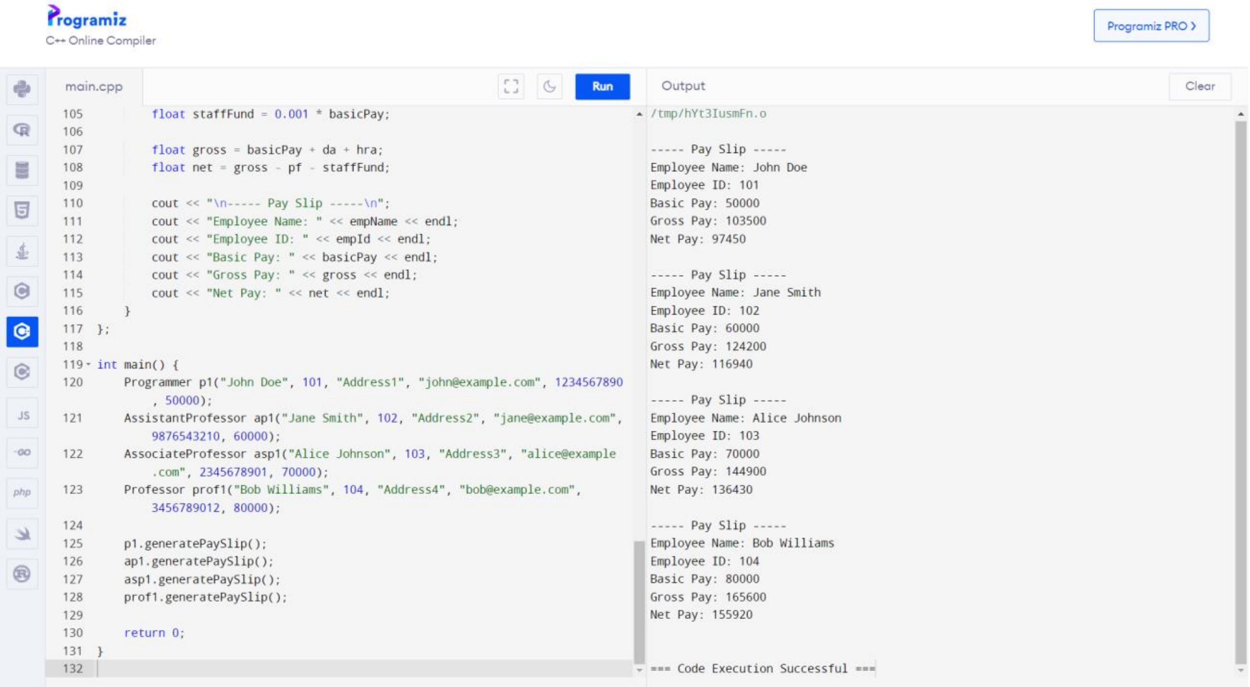


```cpp
    }

    float perimeter() override {
        return 2 * (width + height);
    }
};

class Triangle : public Shape {
public:
    Triangle(float w, float h) : Shape(w, h) {}

    float area() override {
        return 0.5 * width * height;
    }

    float perimeter() override {
        return width + height + sqrt(width * width + height * height);
    }
};

int main() {
    Rectangle rect(5, 4);
    cout << "Area of Rectangle: " << rect.area() << endl;
    cout << "Perimeter of Rectangle: " << rect.perimeter() << endl;

    Triangle tri(3, 4);
    cout << "Area of Triangle: " << tri.area() << endl;
    cout << "Perimeter of Triangle: " << tri.perimeter() << endl;

    return 0;
}

```

```
/tmp/ZtESIivovo.o
Area of Rectangle: 20
Perimeter of Rectangle: 18
Area of Triangle: 6
Perimeter of Triangle: 12


=== Code Execution Successful ===
```

13. Develop a Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of

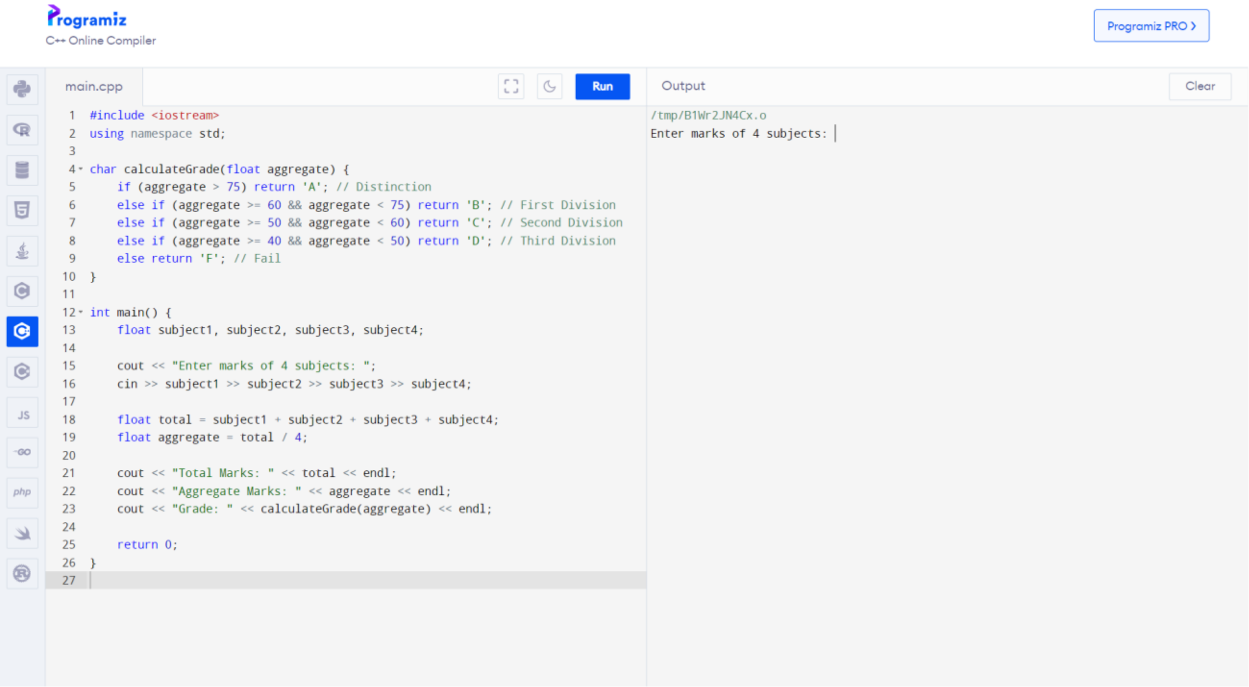
BP for staff club fund. Generate pay slips for the employees with their gross and net salary.



```cpp
        float staffFund = 0.001 * basicPay;

        float gross = basicPay + da + hra;
        float net = gross - pf - staffFund;

        cout << "\n----- Pay Slip -----\n";
        cout << "Employee Name: " << empName << endl;
        cout << "Employee ID: " << empId << endl;
        cout << "Basic Pay: " << basicPay << endl;
        cout << "Gross Pay: " << gross << endl;
        cout << "Net Pay: " << net << endl;
    }
};

int main() {
    Programmer p1("John Doe", 101, "Address1", "john@example.com", 1234567890, 50000);
    AssistantProfessor ap1("Jane Smith", 102, "Address2", "jane@example.com", 9876543210, 60000);
    AssociateProfessor asp1("Alice Johnson", 103, "Address3", "alice@example.com", 2345678901, 70000);
    Professor prof1("Bob Williams", 104, "Address4", "bob@example.com", 3456789012, 80000);

    p1.generatePaySlip();
    ap1.generatePaySlip();
    asp1.generatePaySlip();
    prof1.generatePaySlip();

    return 0;
}
```

```
/tmp/hYt3IusmFn.o

----- Pay Slip -----
Employee Name: John Doe
Employee ID: 101
Basic Pay: 50000
Gross Pay: 103500
Net Pay: 97450

----- Pay Slip -----
Employee Name: Jane Smith
Employee ID: 102
Basic Pay: 60000
Gross Pay: 124200
Net Pay: 116940

----- Pay Slip -----
Employee Name: Alice Johnson
Employee ID: 103
Basic Pay: 70000
Gross Pay: 144900
Net Pay: 136430

----- Pay Slip -----
Employee Name: Bob Williams
Employee ID: 104
Basic Pay: 80000
Gross Pay: 165600
Net Pay: 155920


=== Code Execution Successful ===
```

14. Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50 >= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is Fail.



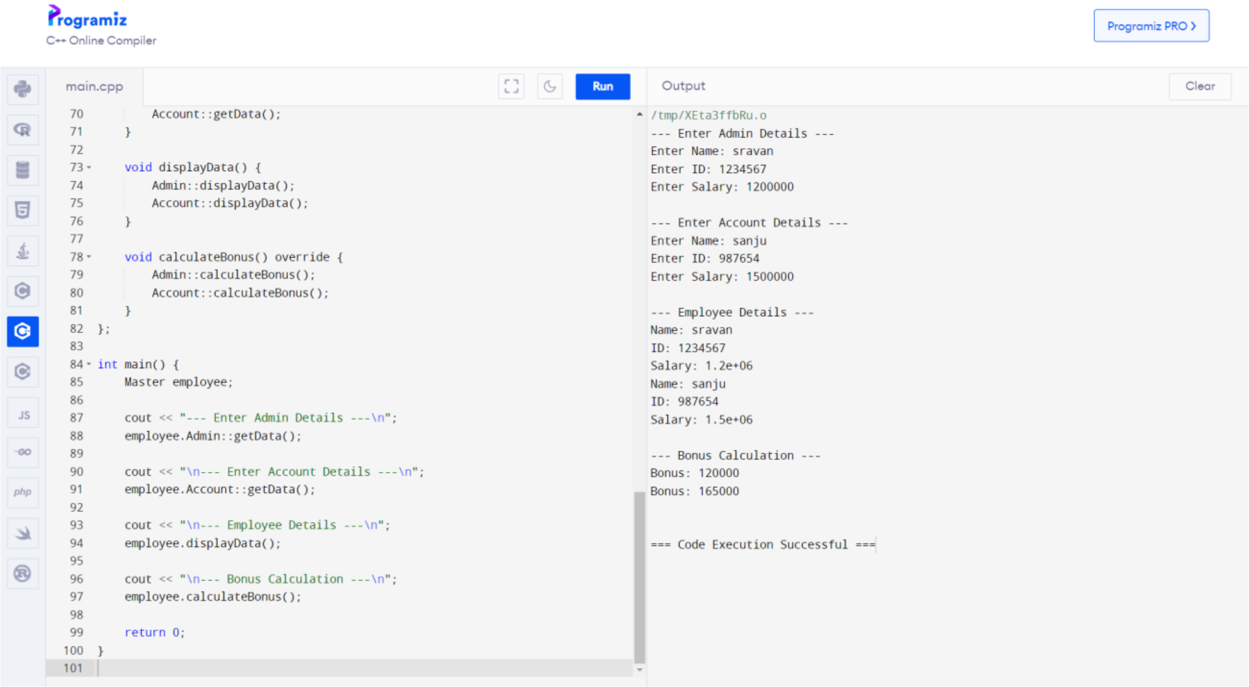
```cpp
#include <iostream>
using namespace std;

char calculateGrade(float aggregate) {
    if (aggregate > 75) return 'A'; // Distinction
    else if (aggregate >= 60 && aggregate < 75) return 'B'; // First Division
    else if (aggregate >= 50 && aggregate < 60) return 'C'; // Second Division
    else if (aggregate >= 40 && aggregate < 50) return 'D'; // Third Division
    else return 'F'; // Fail
}

int main() {
    float subject1, subject2, subject3, subject4;

    cout << "Enter marks of 4 subjects: ";
    cin >> subject1 >> subject2 >> subject3 >> subject4;

    float total = subject1 + subject2 + subject3 + subject4;
    float aggregate = total / 4;

    cout << "Total Marks: " << total << endl;
    cout << "Aggregate Marks: " << aggregate << endl;
    cout << "Grade: " << calculateGrade(aggregate) << endl;

    return 0;
}

```

```
/tmp/B1Wr2JN4Cx.o
Enter marks of 4 subjects: 
```

15. Write a program to calculate the bonus of the employees. The class master derives the information from both admin and account classes which derives information from the class person. Create base and all derived classes having same member functions and parameters called getdata, display data and bonus.

enter salary : 10000

bonus = 11000



```cpp
        Account::getData();
    }

    void displayData() {
        Admin::displayData();
        Account::displayData();
    }

    void calculateBonus() override {
        Admin::calculateBonus();
        Account::calculateBonus();
    }
};

int main() {
    Master employee;

    cout << "--- Enter Admin Details ---\n";
    employee.Admin::getData();

    cout << "\n--- Enter Account Details ---\n";
    employee.Account::getData();

    cout << "\n--- Employee Details ---\n";
    employee.displayData();

    cout << "\n--- Bonus Calculation ---\n";
    employee.calculateBonus();

    return 0;
}

```

```
/tmp/XEta3ffbRu.o
--- Enter Admin Details ---
Enter Name: sravan
Enter ID: 1234567
Enter Salary: 1200000

--- Enter Account Details ---
Enter Name: sanju
Enter ID: 987654
Enter Salary: 1500000

--- Employee Details ---
Name: sravan
ID: 1234567
Salary: 1.2e+06
Name: sanju
ID: 987654
Salary: 1.5e+06

--- Bonus Calculation ---
Bonus: 120000
Bonus: 165000


=== Code Execution Successful ===
```