OOPS C++ PROGRAMS
STUDENT DETAILS USING CLASS AND OBJECTS

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

using namespace std;

class student_details {
private:
    string name;
    int roll, age, marks;

public:
    void read();
    void display();
};

void student_details::read() {
    cout << "Enter name: ";
    cin >> name;
    cout << "Enter roll: ";
    cin >> roll;
    cout << "Enter age: ";
    cin >> age;
    cout << "Enter marks: ";
    cin >> marks;
}

void student_details::display() {
    cout << "Student details are:" << endl;
    cout << "Name: " << name << endl;
    cout << "Roll: " << roll << endl;
    cout << "Age: " << age << endl;
    cout << "Marks: " << marks << endl;
}

int main() {
    student_details student1;

    cout << "Enter details for student 1:" << endl;
    student1.read();

    cout << endl;
```

```
    student1.display();

    return 0;
}
```

OUTPUT

```
Enter details for student 1:
Enter name: Alice
Enter roll: 101
Enter age: 18
Enter marks: 95

Student details are:
Name: Alice
Roll: 101
Age: 18
Marks: 95
```

2) **BOOK DETAILS PROGRAM USING MEMBERFUNCTION OUTSIDE THE CLASS**

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

class Bookdetails {
   int year, cost;
   char title[20], name[10];

public:
   void read();
   void display();
};

void Bookdetails::read() {
   cout << "Enter title, author name: ";
   cin >> title >> name;
   cout << "Enter year and cost: ";
   cin >> year >> cost;
}

void Bookdetails::display() {
   cout << "the Bookdetails are" << endl;
   cout << "book title is " << title << endl;
   cout << "author name is " << name << endl;
   cout << "Year is " << year << endl;
   cout << "Cost is " << cost << endl;
}
```

OUTPUT

```
Enter title, author name: The Great Gatsby F. Scott Fitzgerald
Enter year and cost: 1925 15.99
the Bookdetails are
book title is The
author name is Great
Year is 1925
Cost is 15.99
```

3)EMPLOYEE PAY SLIP PROGRAM

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

class Employee {
private:
   double basicpay, da, hra, grossSalary, tax, netSalary;

public:
   void read() {
      cout << "Enter basic pay: ";
      cin >> basicpay;
   }

   void calculate() {
      da = basicpay * 0.70;
      hra = basicpay * 0.10;
      grossSalary = basicpay + da + hra;
      tax = grossSalary * 0.20;
      netSalary = grossSalary - tax;
   }

   void display() {
      cout << "\nEmployee Payslip" << endl;
      cout << "Basic pay: " << basicpay << endl;
      cout << "DA (70% of Basic): " << da << endl;
      cout << "HRA (10% of Basic): " << hra << endl;
      cout << "Gross Salary: " << grossSalary << endl;
      cout << "Tax (20% of Gross): " << tax << endl;
      cout << "Net Salary: " << netSalary << endl;
   }
};

int main() {
   Employee emp;
   emp.read();
   emp.calculate();
   emp.display();
   return 0;
}
```

OUTPUT

```
Enter basic pay: 10000

Employee Payslip
```

Basic pay: 10000
DA (70% of Basic): 7000
HRA (10% of Basic): 1000
Gross Salary: 18000
Tax (20% of Gross): 3600
Net Salary: 14400

4) ELECTRICITY BILL PROGRAM

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

class ElectricityBill {
    int consumerNo;
    string consumerName;
    int prevReading, currReading, unitsConsumed;
    string connectionType;
    double billAmount;

public:
    void read() {
        cout << "Enter consumer number: ";
        cin >> consumerNo;

        cout << "Enter consumer name: ";
        cin >> consumerName;

        cout << "Enter previous month reading: ";
        cin >> prevReading;

        cout << "Enter current month reading: ";
        cin >> currReading;

        cout << "Enter type of EB connection (domestic/commercial): ";
        cin >> connectionType;

        // Convert to lowercase for uniform comparison
        for (auto &c : connectionType)
            c = tolower(c);

        unitsConsumed = currReading - prevReading;

        if (unitsConsumed < 0) {
            cout << "Error: Current reading cannot be less than previous reading!" << endl;
            unitsConsumed = 0;
        }
    }

    void calculateBill() {
        int units = unitsConsumed;
```

```
        billAmount = 0.0;

        if (connectionType == "domestic") {
            if (units <= 100)
                billAmount = units * 1.0;
            else if (units <= 200)
                billAmount = 100 * 1.0 + (units - 100) * 2.5;
            else if (units <= 500)
                billAmount = 100 * 1.0 + 100 * 2.5 + (units - 200) * 4.0;
            else
                billAmount = 100 * 1.0 + 100 * 2.5 + 300 * 4.0 + (units - 500) * 6.0;
        }

        else if (connectionType == "commercial") {
            if (units <= 100)
                billAmount = units * 2.0;
            else if (units <= 200)
                billAmount = 100 * 2.0 + (units - 100) * 4.5;
            else if (units <= 500)
                billAmount = 100 * 2.0 + 100 * 4.5 + (units - 200) * 6.0;
            else
                billAmount = 100 * 2.0 + 100 * 4.5 + 300 * 6.0 + (units - 500) * 7.0;
        } else {
            cout << "Invalid connection type entered!" << endl;
        }
    }

    void display() {
        cout << "\n----- ELECTRICITY BILL -----" << endl;
        cout << "Consumer No: " << consumerNo << endl;
        cout << "Consumer Name: " << consumerName << endl;
        cout << "Connection Type: " << connectionType << endl;
        cout << "Previous Reading: " << prevReading << endl;
        cout << "Current Reading: " << currReading << endl;
        cout << "Units Consumed: " << unitsConsumed << endl;
        cout << "Total Bill Amount: Rs. " << billAmount << endl;
        cout << "----------------------------" << endl;
    }
};

int main() {
    ElectricityBill eb;
    eb.read();
    eb.calculateBill();
```

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

OUTPUT

```
Enter consumer number: 1022
Enter consumer name: Ananya
Enter previous month reading: 500
Enter current month reading: 750
Enter type of EB connection (domestic/commercial): commercial

----- ELECTRICITY BILL -----
Consumer No: 1022
Consumer Name: Ananya
Connection Type: commercial
Previous Reading: 500
Current Reading: 750
Units Consumed: 250
Total Bill Amount: Rs. 1050
----------------------------
```

5)N STUDENT (ARRAY OF OBJECTS)

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

using namespace std;

class Student {
private:
    string name;
    int roll_number;

public:
    void getDetails() {
        cout << "Enter student name: ";
        cin >> name;
        cout << "Enter roll number: ";
        cin >> roll_number;
    }

    void displayDetails() {
        cout << "Student Name: " << name << endl;
        cout << "Roll Number: " << roll_number << endl;
    }
};

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

    // The code is incomplete, as it does not contain a loop to get or display details for the student
objects.
    // Therefore, a specific output cannot be generated.
    // The program would prompt for the number of students and then terminate.
}
```

OUTPUT

```
Enter the number of students:
<user enters a number, e.g., 2>
Enter student name:
<user enters a name, e.g., John Doe>
Enter roll number:
<user enters a roll number, e.g., 101>
```

Enter student name:
<user enters a name, e.g., Jane Smith>
Enter roll number:
<user enters a roll number, e.g., 102>
Student Name: John Doe
Roll Number: 101
Student Name: Jane Smith
Roll Number: 102

6) DEMONSTRATE THE CONSTRUCTOR AND DESTRUCTOR

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

class Rectangle {
private:
    double length;
    double breadth;

public:
    Rectangle(double l, double b) {
        length = l;
        breadth = b;
        cout << "Constructor called. A new rectangle with length " << length << " and breadth " <<
breadth << " has been created." << endl;
    }

    ~Rectangle() {
        cout << "Destructor called. The rectangle with length " << length << " and breadth " <<
breadth << " is being destroyed." << endl;
    }

    void displayArea() {
        cout << "Area of the rectangle is: " << length * breadth << endl;
    }
};

int main() {
    Rectangle rect1(10.5, 5.2);
    rect1.displayArea();
    return 0;
}
```

OUTPUT

constructor called. A new rect angle length and breadth breadth has b
Destructor called: The rectangle with length and breadth is being destroyed

7) CONSTRUCTOR OVERLOADING

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

using namespace std;

class color {
private:
    int red;
    int green;
    int blue;

public:
    color() {
        red = 0;
        green = 0;
        blue = 0;
        cout << "Default constructor called: created.color." << endl;
    }

    color(int x, int y, int b) {
        red = x;
        green = y;
        blue = b;
        cout << "parameterised constructor called: Created with RGB (" << red << ", " << green <<
", " << blue << ")" << endl;
    }
};

int main() {
    color c1; // Calls the default constructor
    color c2(255, 0, 128); // Calls the parameterized constructor
    return 0;
}
```

OUTPUT

```
Default constructor called: created.
Parameterised constructor called: Created with RGB (255, 0, 128)
```

8) IMPLEMENT OBJECT AS FUNCTION ARGUMENTS

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

class Box {
public:
   double width;
   Box(double w) {
      width = w;
   }
};

bool isLarger(Box b1, Box b2) {
   return b1.width > b2.width;
}

int main() {
   Box box1(10.5);
   Box box2(8.2);

   cout << "Width of the box 1 is: " << box1.width << endl;
   cout << "Width of the box 2 is: " << box2.width << endl;

   if (isLarger(box1, box2)) {
      cout << "box 1 is larger than box 2" << endl;
   } else {
      cout << "box 2 is larger than box 1" << endl;
   }

   return 0;
}
```

OUTPUT

```
Width of the box 1 is: 10.5
Width of the box 2 is: 8.2
box 1 is larger than box 2
```

9) FUNCTION OVERLOADING

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

// Function to add two numbers
int add(int a, int b) {
    return (a + b);
}

// Function to add three numbers
int add(int a, int b, int c) {
    return (a + b + c);
}

// Function to add four numbers
int add(int a, int b, int c, int d) {
    return (a + b + c + d);
}

int main() {
    cout << "Demonstrating Function Overloading" << endl;
    cout << "Sum of 5 and 10 is: " << add(5, 10) << endl;
    cout << "Sum of 5, 10 and 15 is: " << add(5, 10, 15) << endl;
    cout << "Sum of 5, 10, 15, and 20 is: " << add(5, 10, 15, 20) << endl;
    return 0;
}
```

OUTPUT

```
Demonstrating Function Overloading
Sum of 5 and 10 is: 15
Sum of 5, 10 and 15 is: 30
Sum of 5, 10, 15, and 20 is: 50
```

10) INLINE FUNCTION

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

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

int main() {
    int num1, num2;
    cout << "Enter two numbers" << endl;
    cin >> num1 >> num2;
    int sum = add(num1, num2);
    cout << "sum=" << sum << endl;
    return 0;
}
```

OUTPUT

```
Enter two numbers
5 3
sum=8
```