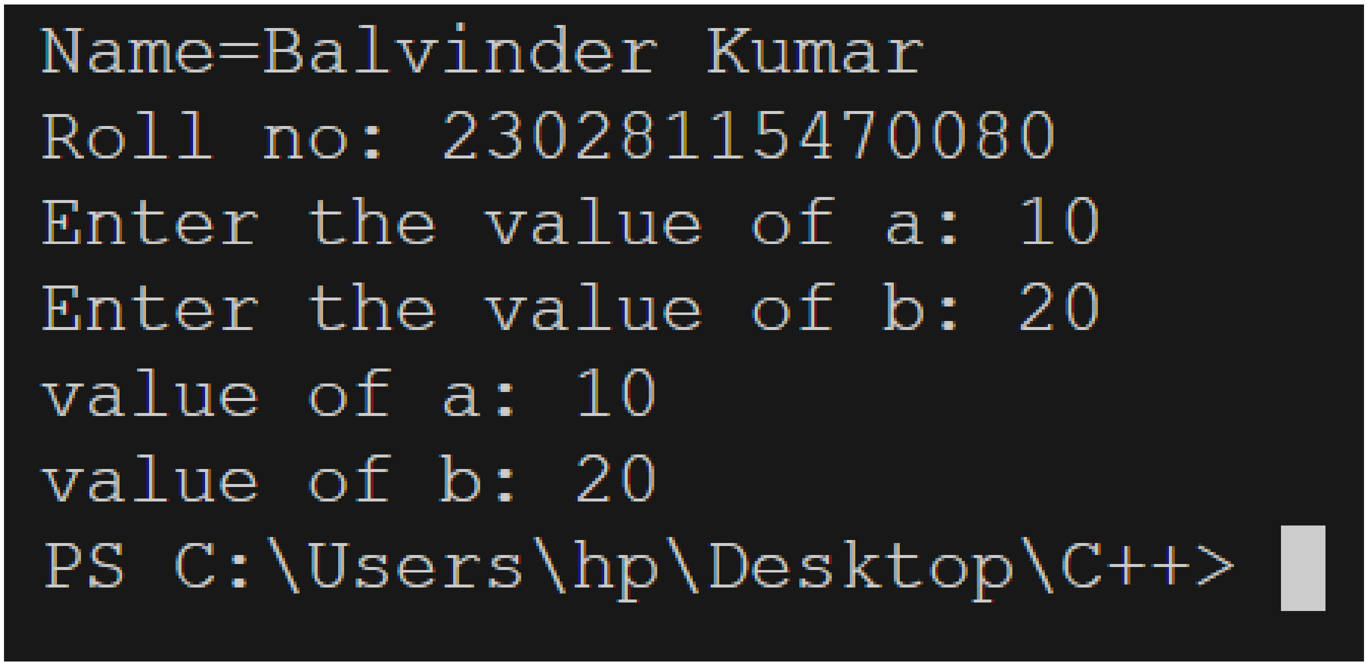
Program 1: Write a program to implement class with function definition inside and outside the class .

Source code:

```
#include <iostream>
using namespace std;
class student {
private:
    int a, b;
    void putdata(); // Declaration
public:
    void getdata() {
        cout << "Enter the value of a: ";
        cin >> a;
        cout << "Enter the value of b: ";
        cin >> b;
        putdata(); // Call putdata within the class
    }
};
void student::putdata() {
    cout << "value of a: " << a << "₩n";
    cout << "value of b: " << b;
}
int main() {
    student s1;
    cout << "Name=Balvinder Kumar₩nRoll no: 23028115470080₩n";
    s1.getdata();
    return 0;
}
```

Output:



```
Name=Balvinder Kumar
Roll no: 23028115470080
Enter the value of a: 10
Enter the value of b: 20
value of a: 10
value of b: 20
PS C:\Users\hp\Desktop\C++>
```

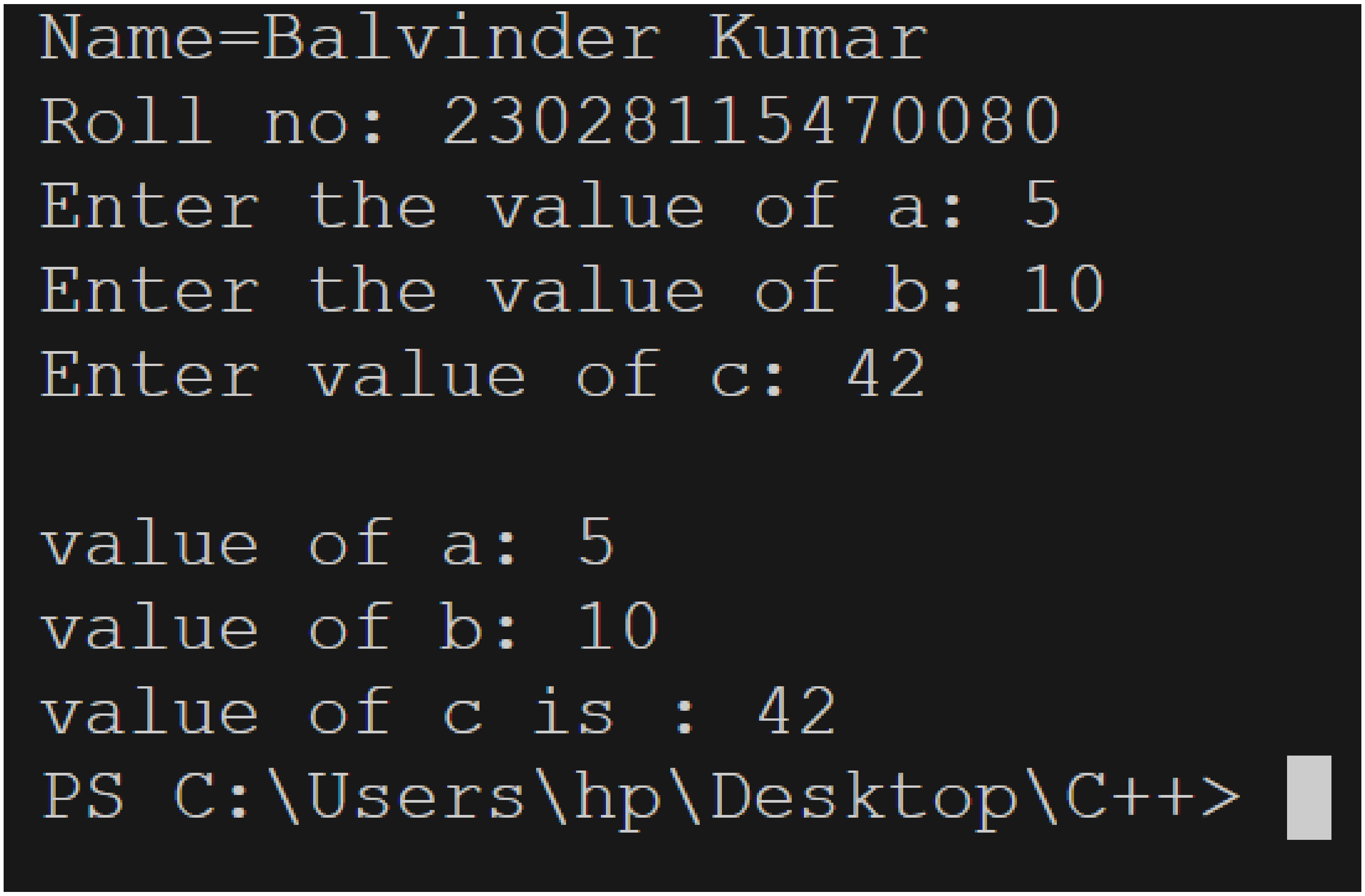
Program 2: Write a program to use static data member and static member function.

Source code:

```
#include <iostream>
class student {
private:
    int a, b;
    static int c;
    void putdata(); // Declaration
public:
    void getdata() {
        std::cout << "Enter the value of a: ";
        std::cin >> a;
        std::cout << "Enter the value of b: ";
        std::cin >> b;
    }
    static void function() {
        std::cout << "Enter value of c: ";
        std::cin >> c;
    }
    static void fun() {
        std::cout << "₩nvalue of c is : " << c;
    }
    void gtdata() {
        putdata();
    }
};
int student::c;
void student::putdata() {
    std::cout << "₩nvalue of a: " << a << "₩n";
    std::cout << "value of b: " << b;
}
```

```
int main() {
    student s1;
    std::cout << "Name=Balvinder Kumar₩nRoll no: 23028115470080₩n";
    s1.getdata();
    student::function();
    s1.gtdata();
    student::fun();
    return 0;
}
```

Output:



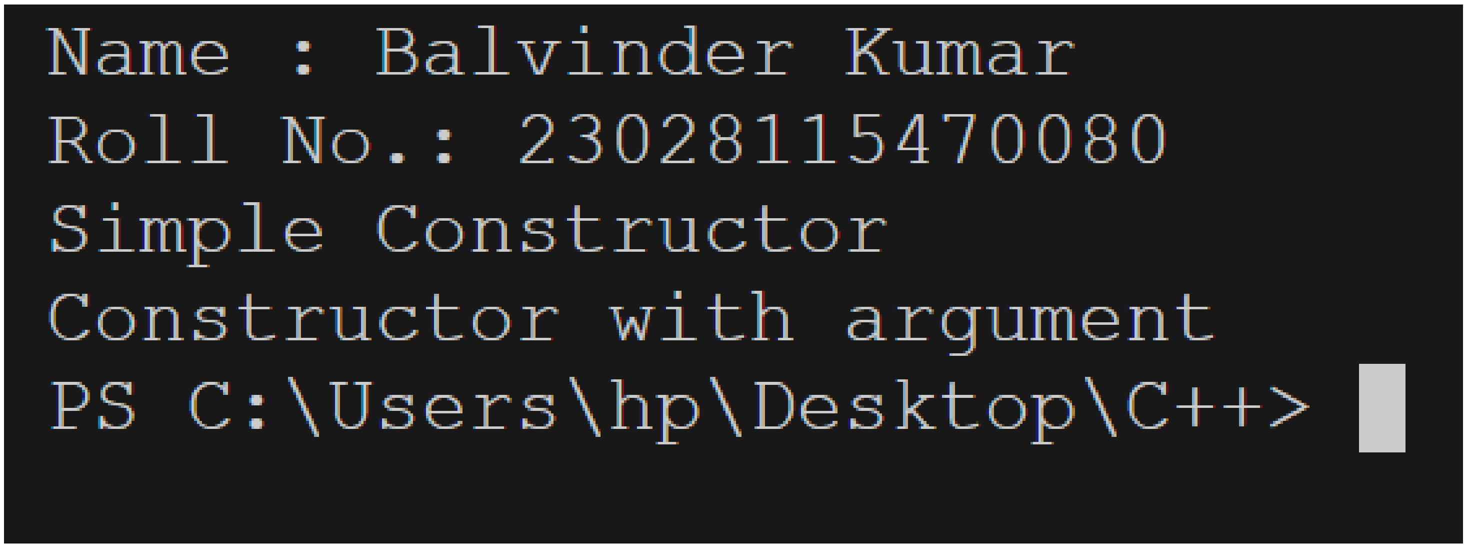
```
Name=Balvinder Kumar
Roll no: 23028115470080
Enter the value of a: 5
Enter the value of b: 10
Enter value of c: 42

value of a: 5
value of b: 10
value of c is : 42
PS C:\Users\hp\Desktop\C++>
```

Program: 3 write a program to implement the constructor with and without argument.

Source code:

```
#include <iostream>
class student {
private:
    int a, b;
public:
    student() {
        std::cout << "Simple Constructor₩n";
    }
    student(int a, int b) {
        std::cout << "Constructor with argument₩n";
    }
};
int main() {
    std::cout << "Name : Balvinder Kumar₩nRoll No.: 23028115470080₩n";
    student a1;
    student b2(0, 0);
    return 0;
}
```

Output:



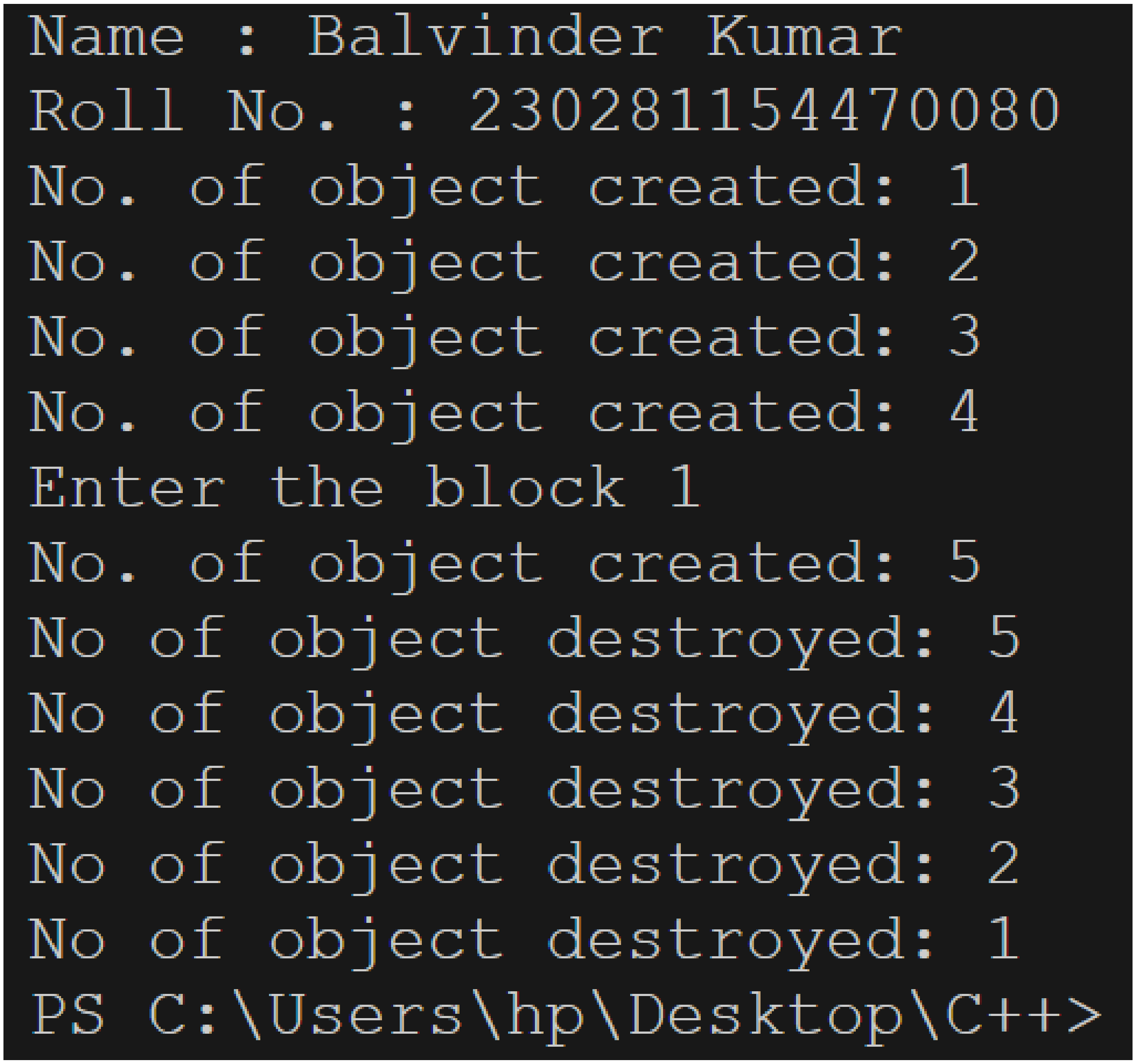
```
Name : Balvinder Kumar
Roll No.: 23028115470080
Simple Constructor
Constructor with argument
PS C:\Users\hp\Desktop\C++>
```

Program: 4 write a program to implement the Destructor.

Source code:

```
#include <iostream>
class student {
private:
    static int count;
public:
    student() {
        count++;
        std::cout << "₩nNo. of object created: " << count;
    }
    ~student() {
        std::cout << "₩nNo of object destroyed: " << count;
        count--;
    }
};
int student::count = 0;
int main() {
    std::cout << "Name : Balvinder Kumar ₩nRoll No. : 230281154470080";
    student R1, R2, R3, R4;
    {
        std::cout << "₩nEnter the block 1";
        student R5;
    }
    return 0;
}
```

Output :



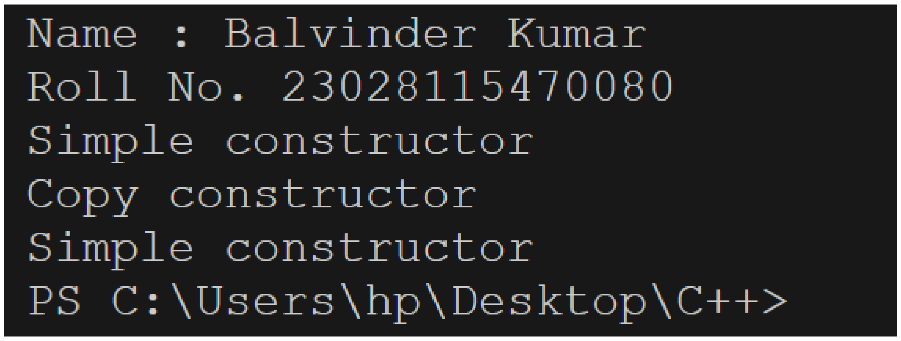
```
Name : Balvinder Kumar
Roll No. : 2302811544 70080
No. of object created: 1
No. of object created: 2
No. of object created: 3
No. of object created: 4
Enter the block 1
No. of object created: 5
No of object destroyed: 5
No of object destroyed: 4
No of object destroyed: 3
No of object destroyed: 2
No of object destroyed: 1
PS C:\Users\hp\Desktop\C++>
```

Program: 5 write a program to implement the copy constructor.

Source code:

```
#include <iostream>

class student {

public:

    student() {

        std::cout << "Simple constructor₩n";

    }

    student(const student &c) {

        std::cout << "Copy constructor₩n";

    }

};

int main() {

    std::cout << "Name : Balvinder Kumar₩nRoll No. 23028115470080₩n";

    student r1, r2(r1);

    student r;

    return 0;

}
```

Output :



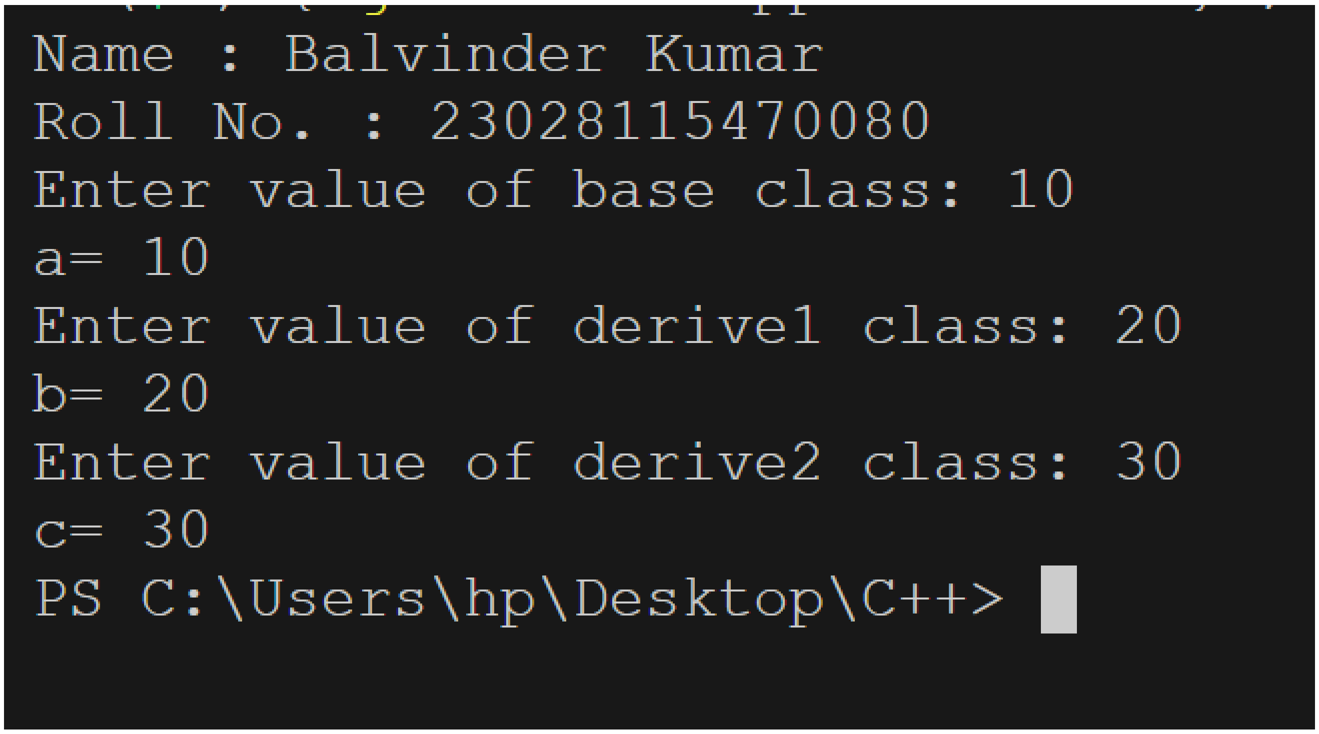
```
Name : Balvinder Kumar
Roll No. 23028115470080
Simple constructor
Copy constructor
Simple constructor
PS C:\Users\hp\Desktop\C++>
```

Program 6: Write a program to implement the Hierarchy Inheritance.

Source Code:

```
#include <iostream>

using namespace std;

class base {

private:

    int a;

public:

    void input() {

        cout << "Enter value of base class: ";

        cin >> a;

    }

    void show() {

        cout << "a= " << a << endl;

    }

};

class derive1 : public base {

private:

    int b;

public:

    void input1() {

        cout << "Enter value of derive1 class: ";

        cin >> b;
    }
    void show1() {
        cout << "b= " << b << endl;
    }
};
class derive2 : public derive1 {
private:
    int c;
public:
    void input2() {
        cout << "Enter value of derive2 class: ";
        cin >> c;
    }
    void show2() {
        cout << "c= " << c << endl;
    }
};
int main() {
    cout << "Name : Balvinder Kumar ₩nRoll No. : 23028115470080₩n";
    derive2 ob2;
    ob2.input();
    ob2.show();
    ob2.input1();
    ob2.show1();

    ob2.input2();

    ob2.show2();

    return 0;

}
```

Output :



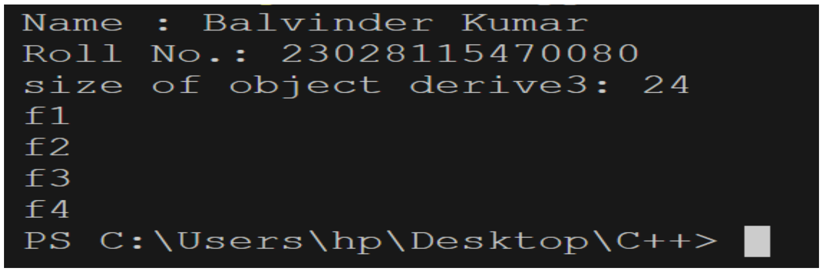
```
Name : Balvinder Kumar
Roll No. : 23028115470080
Enter value of base class: 10
a= 10
Enter value of derive1 class: 20
b= 20
Enter value of derive2 class: 30
c= 30
PS C:\Users\hp\Desktop\C++>
```

Program 7: Write a program to implement the Hybrid Inheritance.

Source Code:

```
#include <iostream>

using namespace std;

class Base {

public:

    int a;

    void f1() {

        cout << "₩nf1"; }};

class derive1 : virtual public Base {

public:

    int a1;

    void f2() {

        cout << "₩nf2";

    }

};

class derive2 : virtual public Base {

public:

    int a2;

    void f3() {

        cout << "₩nf3";

    }
};
class derive3 : public derive1, public derive2 {
public:
    int a3;
    void f4() {
        cout << "₩nf4";
    }
};
int main() {
    derive3 d3;
    cout << "Name : Balvinder Kumar/nRoll No.: 23028115470080/n";
    cout << "size of object derive3: " << sizeof(d3);
    d3.Base::f1(); // Specifying the Base class explicitly
    d3.f2();
    d3.f3();
    d3.f4();
    return 0;
}
```

Outpu :



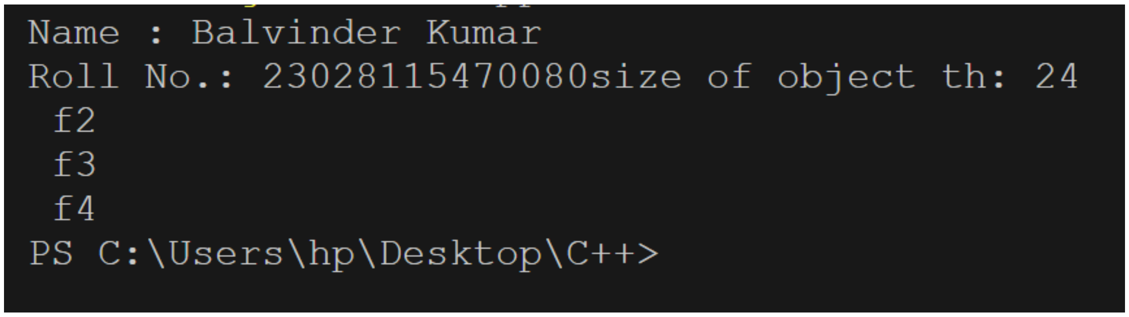
```
Name : Balvinder Kumar
Roll No.: 23028115470080
size of object derive3: 24
f1
f2
f3
f4
PS C:\Users\hp\Desktop\C++>
```

Program 8: Write a program to implement the Diamond Problem.

Source Code:

```
#include <iostream>

class zero {

public:

    int A;

    void f1() {

        std::cout << "₩nf1";

    }

};

class one : virtual public zero {

public:

    int B;

    void f2() {

        std::cout << "₩n f2";

    }

};

class two : virtual public zero {

public:

    int C;

    void f3() {

        std::cout << "₩n f3";

    }
};

class three : public two, public one {

public:

    int D;

    void f4() {

        std::cout << "₩n f4";

    }

};

int main() {

    three th;

    std::cout << "Name : Balvinder Kumar ₩nRoll No.: 23028115470080";

    std::cout << "size of object th: " << sizeof(th);

    th.f2();

    th.f3();

    th.f4();

    return 0;

}
```

Output :



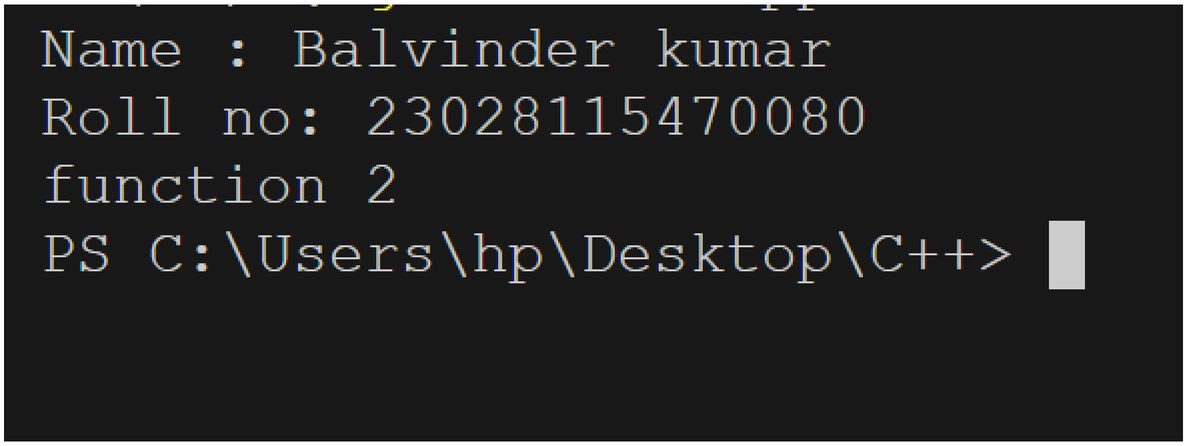
```
Name : Balvinder Kumar
Roll No.: 23028115470080size of object th: 24
 f2
 f3
 f4
PS C:\Users\hp\Desktop\C++>
```

Program 9: Write a program to implement the Virtual function.

Source code:

```
#include <iostream>

class base {

public:

    virtual void f1() = 0; // Pure virtual function

};

class derived : public base {

public:

    virtual void f1() override { // Override keyword

        std::cout << "function 2";

    }

};

int main() {

    base *p;

    derived s2;

    p = &s2;

    std::cout << "Name : Balvinder kumar ₩nRoll no: 23028115470080₩n";

    p->f1();

    return 0;

}
```

Output :



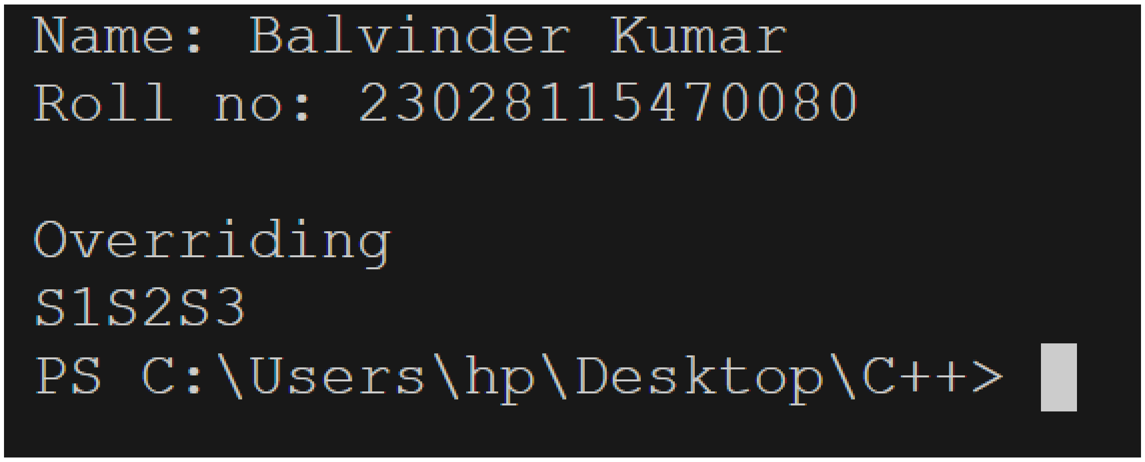
```
Name : Balvinder kumar
Roll no: 23028115470080
function 2
PS C:\Users\hp\Desktop\C++>
```

Program 10: Write a program to implement the Polymorphism.

Source code:

```
#include <iostream>

class A {

public:

    void f1() {

        std::cout << "S1";

    }

};

class B {

public:

    void f1() {

        std::cout << "S2";

    }

};

class C : public A, public B {

public:

    void f1() {

        A::f1(); // Explicitly call A's f1()

        B::f1(); // Explicitly call B's f1()

        std::cout << "S3";

    }

};

int main() {
    std::cout << "Name: Balvinder Kumar /nRoll no: 23028115470080/n";

    std::cout << "/nOverriding₩n";

    C o1;

    o1.f1();

    return 0;

}
```

Output :



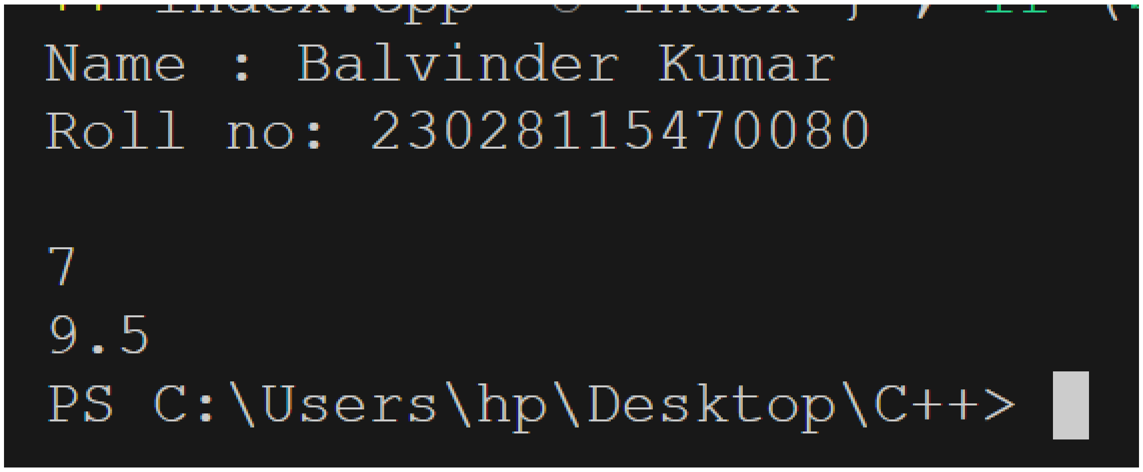
```
Name: Balvinder Kumar
Roll no: 23028115470080

Overriding
S1S2S3
PS C:\Users\hp\Desktop\C++>
```

Program 11: Write a program to implement the Template.

Source code:

```
#include <iostream>

template<class A>

A f1(const A& x, const A& y) {

    if (x > y)

        return x;

    else

        return y;

}

int main() {

    std::cout << "Name : Balvinder Kumar/nRoll no: 23028115470080/n";

    std::cout << "₩n" << f1(4, 7) << "₩n";

    std::cout << f1(9.5, 2.2);

    return 0;

}
```

Output :



```
Name : Balvinder Kumar
Roll no: 23028115470080

7
9.5
PS C:\Users\hp\Desktop\C++>
```