**Program 1:BigInt class overloaded with various operators as required in the assignment**

#include <iostream>

class BigInt {

private:

int num;

public:

BigInt() : num(0) {}

BigInt(int num) : num(num) {}

// Binary operators

BigInt operator+(const BigInt& other) const {

return BigInt(num + other.num);

}

BigInt operator-(const BigInt& other) const {

return BigInt(num - other.num);

}

BigInt operator\*(const BigInt& other) const {

return BigInt(num \* other.num);

}

BigInt operator/(const BigInt& other) const {

if (other.num == 0) {

std::cerr << "Division by zero!" << std::endl;

return BigInt();

}

return BigInt(num / other.num);

}

// Overload operators with one BigInt operand and one int

BigInt operator+(int value) const {

return BigInt(num + value);

}

BigInt operator-(int value) const {

return BigInt(num - value);

}

BigInt operator\*(int value) const {

return BigInt(num \* value);

}

BigInt operator/(int value) const {

if (value == 0) {

std::cerr << "Division by zero!" << std::endl;

return BigInt();

}

return BigInt(num / value);

}

// Unary operators

BigInt& operator++() {

num++;

return \*this;

}

BigInt operator++(int) {

BigInt temp(\*this);

++(\*this);

return temp;

}

BigInt& operator--() {

num--;

return \*this;

}

BigInt operator--(int) {

BigInt temp(\*this);

--(\*this);

return temp;

}

// Display the value

void display() const {

std::cout << "Value: " << num << std::endl;

}

};

int main() {

BigInt a(10);

BigInt b(5);

// Demonstrate binary operators

BigInt result\_add = a + b;

BigInt result\_sub = a - b;

BigInt result\_mul = a \* b;

BigInt result\_div = a / b;

result\_add.display();

result\_sub.display();

result\_mul.display();

result\_div.display();

// Demonstrate unary operators

++a;

a.display();

b++;

b.display();

--a;

a.display();

b--;

b.display();

// Overload operators with one BigInt operand and one int

BigInt c = a + 15;

c.display();

BigInt d = b - 3;

d.display();

BigInt e = a \* 2;

e.display();

BigInt f = b / 2;

f.display();

return 0;

}

**Program 2: Date class with overloaded operators as per assignment**

#include <iostream>

class Date {

private:

int day;

int month;

int year;

public:

Date() : day(1), month(1), year(2000) {}

Date(int d, int m, int y) : day(d), month(m), year(y) {}

// Binary operator + to work with one Date operand and one int

Date operator+(int days) const {

Date result(\*this);

result.day += days;

// Handle days overflow

while (result.day > 31) {

result.day -= 31;

result.month++;

if (result.month > 12) {

result.month = 1;

result.year++;

}

}

return result;

}

// Binary operator - to work with both Date operands and one Date operand and one int

int operator-(const Date& other) const {

int days1 = year \* 365 + month \* 30 + day;

int days2 = other.year \* 365 + other.month \* 30 + other.day;

return days1 - days2;

}

Date operator-(int days) const {

Date result(\*this);

result.day -= days;

// Handle days underflow

while (result.day < 1) {

result.month--;

if (result.month < 1) {

result.month = 12;

result.year--;

}

result.day += 31;

}

return result;

}

// Unary operators

Date& operator++() {

// Increment the date by one day

day++;

if (day > 31) {

day = 1;

month++;

if (month > 12) {

month = 1;

year++;

}

}

return \*this;

}

Date operator++(int) {

Date temp(\*this);

++(\*this);

return temp;

}

Date& operator--() {

// Decrement the date by one day

day--;

if (day < 1) {

month--;

if (month < 1) {

month = 12;

year--;

}

day = 31;

}

return \*this;

}

Date operator--(int) {

Date temp(\*this);

--(\*this);

return temp;

}

// Display the date

void display() const {

std::cout << "Date: " << day << "/" << month << "/" << year << std::endl;

}

};

int main() {

Date date1(10, 5, 2023);

Date date2(20, 5, 2023);

// Demonstrate binary operators

Date result\_add = date1 + 5;

int days\_diff = date2 - date1;

Date result\_sub = date2 - 10;

date1.display();

result\_add.display();

std::cout << "Days difference: " << days\_diff << std::endl;

result\_sub.display();

// Demonstrate unary operators

++date1;

date1.display();

date2++;

date2.display();

--date1;

date1.display();

date2--;

date2.display();

return 0;

}

**Output Snip:**