```cpp
#include <iostream>
#include <string>
using namespace std;
class Rational
{
private:
    int numer;
    int denom;
public:
    int getNumer() const;
    int getDenom() const;
    void setNumer(int);
    void setDenom(int);
    void input();
    void output() const;
    Rational();
    Rational(int, int = 1);
    void reduce();
    friend Rational operator+(const Rational& a, const Rational& b);
    friend istream& operator>>(istream& strm, Rational& obj);
};
void Rational::reduce()
{
    int x = abs(numer);
    int y = abs(denom);
    // find minimum of x and y
    int min = x;
    if (y < x)
        min = y;

    // finding a common factor greater than 1
    int gcf = 1;
    for (int i = 2; i <= min; i++) {
        if (x % i == 0 && y % i == 0) {
            gcf = i;
        }
    }
    numer = numer / gcf;
    denom = denom / gcf;
    if (denom < 0)
    {
        numer = -numer;
        denom = -denom;
    }
}
Rational::Rational()
{
    numer = 0;
    denom = 1;
```

```
50  }
51  Rational::Rational(int x, int y)
52  {
53      numer = x;
54      if (y != 0)
55          denom = y;
56      else
57          denom = 1;
58      reduce();
59  }
60  int Rational::getNumer() const
61  {
62      return numer;
63  }
64  int Rational::getDenom() const
65  {
66      return denom;
67  }
68  void Rational::setNumer(int x)
69  {
70      numer = x;
71      reduce();
72  }
73  void Rational::setDenom(int x)
74  {
75      denom = x;
76      if (denom == 0)
77          denom = 1;
78      reduce();
79  }
80  void Rational::input()
81  {
82      cout << "Numerator? ";
83      cin >> numer;
84      cout << "Denominator? ";
85      cin >> denom;
86      while (denom == 0)
87      {
88          cout << "Denominator can't be zero!\n";
89          cout << "Denominator? ";
90          cin >> denom;
91      }
92      reduce();
93  }
94  void Rational::output() const
95  {
96      if (denom != 1)
97          cout << numer << "/" << denom << endl;
98      else
```

```
 99          cout << numer << endl;
100  }
101  Rational operator+(const Rational &a, const Rational &b)
102  {
103      Rational c;
104      c.numer = a.numer * b.denom + a.numer * b.denom;
105      c.setDenom(a.getDenom() * b.getDenom());
106      c.reduce();
107      return c;
108  }
109  Rational operator-(const Rational& a, const Rational& b)
110  {
111      int x = a.getNumer() * b.getDenom() - a.getDenom() * b.getNumer();
112      int y= a.getDenom() * b.getDenom();
113      return Rational(x,y);
114  }
115  Rational operator*(const Rational& a, const Rational& b)
116  {
117      Rational c;
118      c.setNumer(a.getNumer() * b.getNumer());
119      c.setDenom(a.getDenom() * b.getDenom());
120      c.reduce();
121      return c;
122  }
123  Rational operator/(const Rational& a, const Rational& b)
124  {
125      Rational c;
126      c.setNumer(a.getNumer() * b.getDenom());
127      c.setDenom(a.getDenom() * b.getNumer());
128      c.reduce();
129      return c;
130  }
131  void operator+=(Rational& a, const Rational& b)
132  {
133      a = a + b;
134  }
135  void operator-=(Rational& a, const Rational& b)
136  {
137      Rational c;
138      c.setNumer(a.getNumer() * b.getDenom() - a.getDenom() * b.getNumer());
139      c.setDenom(a.getDenom() * b.getDenom());
140      c.reduce();
141      a = c;
142  }
143  void operator*=(Rational& a, const Rational& b)
144  {
145      Rational c;
146      c.setNumer(a.getNumer() * b.getNumer());
147      c.setDenom(a.getDenom() * b.getDenom());
```

```
148     c.reduce();
149     a = c;
150 }
151 void operator/=(Rational& a, const Rational& b)
152 {
153     Rational c;
154     c.setNumer(a.getNumer() * b.getDenom());
155     c.setDenom(a.getDenom() * b.getNumer());
156     c.reduce();
157     a = c;
158 }
159 bool operator<(const Rational& a, const Rational& b)
160 {
161     return (a.getNumer() * b.getDenom()) < (a.getDenom() * b.getNumer());
162 }
163 bool operator<=(const Rational& a, const Rational& b)
164 {
165     return (a.getNumer() * b.getDenom()) <= (a.getDenom() * b.getNumer());
166 }
167 bool operator>(const Rational& a, const Rational& b)
168 {
169     return (a.getNumer() * b.getDenom()) > (a.getDenom() * b.getNumer());
170 }
171 bool operator>=(const Rational& a, const Rational& b)
172 {
173     return (a.getNumer() * b.getDenom()) >= (a.getDenom() * b.getNumer());
174 }
175 bool operator==(const Rational& a, const Rational& b)
176 {
177     return (a.getNumer() * b.getDenom()) == (a.getDenom() * b.getNumer());
178 }
179 bool operator!=(const Rational& a, const Rational& b)
180 {
181     return (a.getNumer() * b.getDenom()) != (a.getDenom() * b.getNumer());
182 }
183 Rational operator++(Rational& a) // prefix ++x
184 {
185     a.setNumer(a.getNumer() + a.getDenom());
186     return a;
187 }
188 Rational operator++(Rational& a, int n) // postfix x++
189 {
190     Rational b = a;
191     a.setNumer(a.getNumer() + a.getDenom());
192     return b;
193 }
194 ostream& operator<<(ostream& strm, const Rational& obj)
195 {
196     if (obj.getDenom() != 1)
```

```
197         strm << obj.getNumer() << "/" << obj.getDenom();
198     else
199         strm << obj.getNumer();
200     return strm;
201 }
202 istream& operator>>(istream& strm, Rational& obj)
203 {
204     cout << "Numerator? ";
205     strm >> obj.numer;
206     cout << "Denominator? ";
207     strm >> obj.denom;
208     while (obj.denom == 0)
209     {
210         cout << "Denominator can't be zero!\n";
211         cout << "Denominator? ";
212         strm >> obj.denom;
213     }
214     obj.reduce();
215     return strm;
216 }
217 int main()
218 {
219     Rational a(6, -8);
220 
221     cout << a << " is the Rational number" << endl;;
222     cin >> a;
223     cout << a << " is the Rational number";
224 
225     return 0;
226 }
```