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

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

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

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

```
197 istream& operator>>(istream& strm, Rational& obj)
198 {
199     cout << "Numerator? ";
200     strm >> obj.numer;
201     cout << "Denominator? ";
202     strm >> obj.denom;
203     while (obj.denom == 0)
204     {
205         cout << "Denominator can't be zero!\n";
206         cout << "Denominator? ";
207         strm >> obj.denom;
208     }
209     obj.reduce();
210     return strm;
211 }
212 template<typename G>
213 G total(G arr[], int size) {
214     G total = 0;
215     for (int i = 0; i < size; i++)
216     {
217         cout << "please enter a number: ";
218         cin >> arr[i];
219         total += arr[i];
220     }
221     cout<< "These are the numbers you entered:\n";
222     for (int i = 0; i < size; i++)
223         cout << arr[i]<<"\t";
224     cout << endl;
225     return total;
226 }
227 int main()
228 {
229     int* list1 = new int[5];
230     int iTotal = total(list1, 5);
231     cout << "\nSum of the numbers = " << iTotal << endl << endl;
232 
233     double* list2 = new double[4];
234     double dTotal = total(list2, 4);
235     cout << "\nSum of the numbers = " << dTotal << endl << endl;
236 
237     Rational* list3 = new Rational[3];
238     Rational rTotal = total(list3, 3);
239     cout << "\nSum of the numbers = " << rTotal << endl << endl;
240     return 0;
241 };
242 
```