Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/progtech/

Προγραμματιστικές Τεχνικές

Άσκηση 0 Υπόλοιπα πρώτου εξαμήνου

Προθεσμία υποβολής στον grader: 6/3/2020

Ολοκληρώστε την 8η σειρά ασκήσεων του πρώτου εξαμήνου, αν δεν το έχετε ήδη κάνει.

Άσκηση 1 Ρητοί αριθμοί για αρχάριους

Προθεσμία υποβολής στον grader: 13/3/2020

Υλοποιήστε την κλάση `rational` για την αναπαράσταση των *ρητών αριθμών*. Η κλάση αυτή πρέπει να υποστηρίζει τις εξής λειτουργίες:

```
class rational {
public:
  rational (int n, int d);

  rational add (rational r);
  rational sub (rational r);
  rational mul (rational r);
  rational div (rational r);

  void print ();

private:
  int nom, den;  // nominator, denominator

  static int gcd (int a, int b);
};
```

Γράψτε τον παραπάνω ορισμό της κλάσης σε ένα *αρχείο επικεφαλίδας* με όνομα `babyratio.hpp`. Στη συνέχεια, γράψτε τα παρακάτω σε ένα δεύτερο αρχείο με όνομα `babyratio.cpp` και συμπληρώστε κατάλληλα τις υλοποιήσεις των μεθόδων:

```
#ifndef CONTEST
#include "babyratio.hpp"
#endif

using namespace std;

rational::rational (int n, int d) { ... }

rational rational::add (rational r) { ... }

...
```

Προσέξτε ότι οι αριθμητικές πράξεις δεν πρέπει να αλλάζουν το αντικείμενο για το οποίο καλούνται αλλά πρέπει να επιστρέφουν το αποτέλεσμα ως ένα νέο αντικείμενο.

Σημείωση: οι γραμμές "`#ifndef CONTEST`" και "`#endif`" θα σας είναι απαραίτητες για να υποβάλετε τη λύση σας στον grader. Κατά την υποβολή σας, θα σας χρειαστεί μόνο το αρχείο `babyratio.cpp` με τις υλοποιήσεις των μεθόδων, χωρίς τη δήλωση της κλάσης.

Προκειμένου να ελέγξετε την υλοποίησή σας, πριν την υποβάλετε, μπορείτε να χρησιμοποιήσετε ένα κύριο πρόγραμμα, το οποίο θα γράψετε σε ένα τρίτο αρχείο με όνομα `babyratiotest.cpp`. Χρησιμοποιήστε το παρακάτω σαν υπόδειγμα, προσθέστε όμως περισσότερους ελέγχους:

```
#include <iostream>
#include "babyratio.hpp"

using namespace std;

int main() {
  rational a(1, 2);
  rational b(3, 4);
  rational c(5, 6);
  a.add(b).sub(c).print();
  cout << endl;
  a.print();
  cout << " should still be 1/2" << endl;
}
```

Στο περιβάλλον του εργαστηρίου (`novice.softlab.ntua.gr`), μεταγλωττίστε και εκτελέστε το πρόγραμμά σας γράφοντας:

```
pt99a999@novice$ c++ babyratiotest.cpp babyratio.cpp
File compiled successfully.
pt99a999@novice$ run babyratiotest.exec
5/12
1/2 should still be 1/2

--------------------------------------------------------------------------------
Program finished with status 0
Total CPU time: 0.00 seconds
Total memory: 912 KB
```

Άσκηση 2 Ρητοί αριθμοί για προχωρημένους

Προθεσμία υποβολής στον grader: 20/3/2020

Ομοίως με την πρώτη άσκηση, υλοποιήστε την κλάση `rational` για την αναπαράσταση των *ρητών αριθμών*. Η κλάση αυτή πρέπει όμως τώρα να υποστηρίζει τις λειτουργίες που αναγράφονται στο αρχείο επικεφαλίδας `fullratio.hpp`, τα περιεχόμενα του οποίου είναι τα ακόλουθα:

```
#ifndef __FULLRATIO_HPP__
#define __FULLRATIO_HPP__

#include <iostream>

```

```
class rational {
public:
  rational (int n, int d = 1);

  friend rational operator + (const rational &x, const rational &y);
  friend rational operator - (const rational &x, const rational &y);
  friend rational operator * (const rational &x, const rational &y);
  friend rational operator / (const rational &x, const rational &y);

  friend std::ostream & operator << (std::ostream &out, const rational &x);

private:
  int nom, den;

  static int gcd (int a, int b);
};

#endif
```

Όπως και προηγουμένως, γράψτε τα παρακάτω σε ένα δεύτερο αρχείο με όνομα `fullratio.cpp` και συμπληρώστε κατάλληλα τις υλοποιήσεις των μεθόδων:

```
#include <iostream>
#ifndef CONTEST
#include "fullratio.hpp"
#endif

using namespace std;

rational::rational (int n, int d) { ... }

rational operator + (const rational &x, const rational &y) { ... }

...
```

Προκειμένου να ελέγξετε την υλοποίησή σας, πριν την υποβάλετε, μπορείτε να χρησιμοποιήσετε ένα κύριο πρόγραμμα, το οποίο θα γράψετε σε ένα τρίτο αρχείο με όνομα `fullratiotest.cpp`. Χρησιμοποιήστε το παρακάτω σαν υπόδειγμα, προσθέστε όμως περισσότερους ελέγχους:

```
#include <iostream>
#include "fullratio.hpp"

using namespace std;

int main() {
  rational a(1, 2);
  rational b(3, 4);
  rational c(5, 6);
  cout << a + b - c << endl;
  cout << a << " should still be 1/2" << endl;
}
```

Άσκηση 3 Στοίβες με πίνακα

Προθεσμία υποβολής στον grader: 20/3/2020

Υλοποιήστε τον ΑΤΔ `stack` με χρήση **`template`**, έτσι ώστε να μπορείτε να κατασκευάζετε στοίβες ακεραίων αριθμών, στοίβες συμβολοσειρών, και γενικά στοίβες από στοιχεία οποιουδήποτε τύπου `T`. Συμπληρώστε κατάλληλα τα παρακάτω, που θα τοποθετήσετε σε ένα αρχείο με όνομα `stackdemo.cpp`:

```
#include <iostream>

using namespace std;

template <typename T>
class stack {
public:
  stack (int size)                                          { ... }
  stack (const stack &s)                                    { ... }
  ~stack ()                                                 { ... }
  const stack & operator = (const stack &s)                 { ... }

  bool empty ()                                             { ... }
  void push (const T &x)                                    { ... }
  T pop ()                                                  { ... }
  int size ()                                               { ... }

  friend ostream & operator << (ostream &out, const stack &s) { ... }

private:
  ...
};
```

Η υλοποίηση της στοίβας μπορεί να γίνει με χρήση πίνακα και σε αυτή την άσκηση δεν χρειάζεται να υλοποιήσετε ελέγχους στις μεθόδους `push` και `pop`.

Προκειμένου να ελέγξετε την υλοποίησή σας, πριν την υποβάλετε, μπορείτε να χρησιμοποιήσετε ένα κύριο πρόγραμμα σαν το παρακάτω, το οποίο θα προσθέσετε στο τέλος του αρχείου `stackdemo.cpp`:

```
#ifndef CONTEST
int main () {
  // let's play with integers...
  stack<int> s(10);
  cout << "s is empty: " << s << endl;
  s.push(42);
  cout << "s has one element: " << s << endl;
  s.push(17);
  s.push(34);
  cout << "s has more elements: " << s << endl;
  cout << "How many?  " << s.size() << endl;
  stack<int> t(5);
  t.push(7);
  cout << "t: " << t << endl;
  t = s;
  cout << "popping from s: " << s.pop() << endl;
```

```
  s.push(8);
  stack<int> a(s);
  t.push(99);
  a.push(77);
  cout << "s: " << s << endl;
  cout << "t: " << t << endl;
  cout << "a: " << a << endl;
  // now with doubles...
  stack<double> c(4);
  c.push(3.14);
  c.push(1.414);
  cout << "c contains doubles " << c << endl;
  // and with characters...
  stack<char> k(4);
  k.push('$');
  cout << "k contains a character " << k << endl;
}
#endif
```

Στο περιβάλλον του εργαστηρίου (`novice.softlab.ntua.gr`), μεταγλωττίστε και εκτελέστε το πρόγραμμά σας γράφοντας:

```
pt99a999@novice$ c++ stackdemo.cpp
File compiled successfully.
pt99a999@novice$ run stackdemo.exec
s is empty: []
s has one element: [42]
s has more elements: [42, 17, 34]
How many?  3
t: [7]
popping from s: 34
s: [42, 17, 8]
t: [42, 17, 34, 99]
a: [42, 17, 8, 77]
c contains doubles [3.14, 1.414]
k contains a character [$]

--------------------------------------------------------------------------------
Program finished with status 0
Total CPU time: 0.00 seconds
Total memory: 1004 KB
```

Βεβαιωθείτε ότι ο τρόπος που εκτυπώνετε τις στοίβες είναι αυτός που φαίνεται στο παραπάνω παράδειγμα.