Maths 1262 quizzes

Wednesday 12/3/14 answers, due Tuesday 18/3/14 **Your answers should show all work.**

(1: 20 marks) Write code for enough of the following class member functions to allow the main program below to work (g++ does not require unused member functions to be fully coded).

```
typedef class Fraction
{ public:
    Fraction();
    Fraction( int m, int n);
    bool operator == ( Fraction other );
    Fraction operator - (); // unary minus, prefix minus
    Fraction operator + ( Fraction other );
    Fraction operator * ( Fraction other );
    int num();
    int denom();
    double to_double ();
    void print();
  private:
    int nu, de; // numerator and denominator, respectively
} Fraction;
```

Be careful using `nu, de`, which are private. Note $a/b = c/d \iff ad = bc$. The main routine:

```
int main()
{ Fraction a(1,2);
  Fraction b(3,4);
  Fraction c(20,16);
  bool cond;
  (a+b).print();
  cout << endl;
  cond = a+b == c;
  cout << "condition ... " << cond << endl;
  return 0;
}
```

Answer.

```
#include <iostream>
#include <cstdlib>

using namespace std;

typedef class Fraction
{
  public:
    Fraction();
    Fraction( int m, int n);
    bool operator == ( Fraction other );
    Fraction operator - ();
    Fraction operator + ( Fraction other );
    Fraction operator * ( Fraction other );
    int num();
    int denom();
```

```
    double to_double();
    void print();

  private:
    int nu, de; // numerator and denominator, respectively
} Fraction;

Fraction::Fraction()
{
  nu = 0; de = 1;
}

Fraction::Fraction ( int m, int n )
{
  if ( n == 0 )
  {
    cout << "ERROR constructor with zero denominator, abort\n";
    exit (  -1 );
  }
  if ( n<0 )
  {
    n = -n; m = -m;
  }

  nu = m; de = n;
}

bool Fraction::operator== ( Fraction other )
{
  return nu * other.denom() == de * other.num();
}

Fraction Fraction::operator - ()
{
  return Fraction ( -nu, de );
}

int Fraction::num()
{
  return nu;
}

int Fraction::denom()
{
  return de;
}

int gcd ( int m, int n )
{
  int x,y,z;
  if ( m<0 ) m = -m;
  if ( n<0 ) n = -n;

  x = m; y = n;
```

```
  while ( y > 0 )
  {
    z = x % y; x = y; y = z;
  }
  return x;
}

Fraction Fraction::operator + ( Fraction other )
{
  int r, s,g;
  r = de * other.num() + nu * other.denom();
  s = de * other.denom ();
  g = gcd ( r,s );
  return Fraction ( r/g, s/g );
}

void Fraction::print()
{
  cout << nu << '/' << de;
}


int main()
{
  Fraction a(1,2);
  Fraction b(3,4);
  Fraction c(20,16);
  bool cond;
  (a+b).print();
  cout << endl;
  cond = a+b == c;
  cout << "condition ... " << cond << endl;
  return 0;
}
```

(2: 10 marks) There is a potential for trouble in the 'replace word' program, replace.cpp on the course web page. Identify it, and correct it. (You may need to consult the C++ reference for information about strings.) The part of interest is

---

```
while ( ! finished )
  { cin.getline ( buffer, 200 );
    if ( cin.eof() )
      finished = true;
    else
    { str = string ( buffer ); // convert
      int i = str.find ( oldword );
      while ( i < string::npos )
      { str.replace ( i, oldword.length(), newword );
        i = str.find ( oldword );
      }
      cout << str << endl;
    }
  }
```

Answer. If a word is replaced with itself, you can get an infinite loop. One way to correct it is

```
while ( ! finished )
 {
   cin.getline ( buffer, 200 );
   if ( cin.eof() )
     finished = true;
   else
   {
     str = string ( buffer ); // convert
     if ( newword != oldword )
     {
       int i = str.find ( oldword );
       while ( i < string::npos )
       {
         str.replace ( i, oldword.length(), newword );
         i = str.find ( oldword );
       }
     }
     cout << str << endl;
   }
 }
```

(3: 10 marks) Say what's wrong with the following code, and write a proper version.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{ vector<string> v; char buffer[100];
  cin >> buffer;
  while ( !cin.eof() )
  { v.push_back ( string( buffer ) ); cin >> buffer;
  }
  sort ( v.begin(), v.end () );

  for ( int i = 0; i<v.size(); ++i )
    cout << v[i] << endl;
  return 0;
}
```

Answer. This is dangerous because of uncontrolled reading into an array. Just alter one declaration

```
  string buffer;
```

No other change is necessary.
(4: 10 marks) The following code was introduced in class (but it has been slightly altered). It is on the web page.

```
  online = 0;

  for ( vector<string>::reverse_iterator i = v.rbegin();
        i != v.rend(); ++ i )
  { if ( online > 0 )
    {
```

```
      if (( *i ).length() + online + 1 >= 60 )
      {
        cout << endl;
        online = 0;
      }
    }
    if ( online > 0 )
    { cout << ' '; ++ online;
    }
    cout << *i;
    online += (*i).length();
  }

  cout << endl;
```

What happens with long words, i.e., blank-separated input strings containing ≥ 60 alphanumeric characters?

Answer. They are output beginning on their own line, but they are not shortened nor split. That is, long words stay long.