CSE 241

PS #3

11.02.2020

Q1)

Define a class for a type called Fraction. This class is used to represent a ratio of two integers. Include mutator functions that allow the user to set the numerator and the denominator. Also include a member function that returns the value of the numerator divided by the denominator as a double. Include an additional member function that outputs the value of the fraction reduced to lowest terms. For example, instead of outputting 20/60 the function should output 1/3. This will require finding the greatest common divisor for the numerator and denominator, and then dividing both by that number. Embed your class in a test program

Q2)

Define a class called Odometer that will track fuel and mileage for an automotive vehicle. The class should have member variables to track the miles driven and the fuel efficiency of the vehicle in miles per gallon. Include a mutator function to reset the odometer to zero miles, a mutator function to set the fuel efficiency, a mutator function that accepts miles driven for a trip and adds it to the odometer's total, and an accessor function that returns the number of gallons of gasoline that the vehicle has consumed since the odometer was last reset. Use your class with a test program that creates several trips with different fuel efficiencies. You should decide which variables should be public, if any.

Q3)

Define a class named Money that stores a monetary amount. The class should have two private integer variables, one to store the number of dollars and another to store the number of cents. Add accessor and mutator functions to read and set both member variables. Add another function that returns the monetary amount as a double. Write a program that tests all of your functions with at least two different Money objects.

Q4)

Write a function that determines if two strings are anagrams. The function should not be case sensitive and should disregard any punctuation or spaces. Two strings are anagrams if the letters can be rearranged to form each other. For example, "Eleven plus two" is an anagram of "Twelve plus one". Each string contains one "v", three "e's", two "l's", etc. Test your function with several strings that are anagrams and non-anagrams. You may use either the string class or a C-style string.

Q5)

Create a file and then write some text into then file, after writing text file will be closed and again file will open in read mode, read all written text.

A1)

/fraction.cpp

```
//This program defines a class for fractions, which stores a numerator
//  and denominator. A member functions allows for retrieval of the
//  fraction as a double and another outputs the value in lowest
//  terms.
//The greatest common divisor is found to reduce the fraction.
//  In this case we use a brute-force method to find the gcd, but
//  a more efficient solution such as euclid's method could also be
//  used.
#include <iostream>
#include <cstdlib>
using namespace std;
class Fraction
{
public:
double getDouble();
void outputReducedFraction();
void setNumerator(int n);
void setDenominator(int d);
private:
int numerator;
int denominator;
int gcd(); // Finds greatest common divisor of numerator
// and denominator
};
// ---------------------------------
// ----- ENTER YOUR CODE HERE -----
// ---------------------------------
// ======================
// Fraction::getDouble
// Returns the fraction's value as a double
// ======================
```

```
double Fraction::getDouble()
{
return (static_cast<double>(numerator) / denominator);
}
// ======================
// Fraction::outputReducedFraction
// Reduces the fraction and outputs it to the console.
// ======================
void Fraction::outputReducedFraction()
{
 int g;

 g = gcd();
 cout << numerator / g << " / " << denominator / g << endl;
 return;
}
// ======================
// Fraction::setNumerator
// Mutator to change the numerator
// ======================
void Fraction::setNumerator(int n)
{
numerator = n;
}
// ======================
// Fraction::setDenominator
// Mutator to change the denominator
// ======================
void Fraction::setDenominator(int d)
{
denominator = d;
```

```
}
// ======================
// Fraction::gcd
// Finds greatest common divisor of numerator and denominator
// by brute force (start at larger of two, work way down to 1)
// ======================
int Fraction::gcd()
{
  int g;  // candidate for gcd, start at the smaller of the
     // numerator and denominator
  if (numerator > denominator)
  {
g = denominator;
  }
  else
  {
  g = numerator;
  }
  // Work down to 1, testing to see if both numerator and denominator
  // can be divided by g.  If so, return it.
  while (g>1)
  {
  if (((numerator % g)==0) && ((denominator % g)==0))
  return g;
  g--;
  }
  return 1;
}
// --------------------------------
// --------- END USER CODE --------
// --------------------------------
```

```
// ======================
//     main function
// ======================
int main()
{
  // Some test fractions
  Fraction f1, f2;
  f1.setNumerator(4);
  f1.setDenominator(2);
  cout << f1.getDouble() << endl;
  f1.outputReducedFraction();
  cout << endl;
f2.setNumerator(20);
  f2.setDenominator(60);
  cout << f2.getDouble() << endl;
  f2.outputReducedFraction();
  cout << endl;
}
```

A2)

```
//odometer.cpp
//This program defines an Odometer class to track fuel and mileage.
//  It stores the miles driven and fuel efficient and includes
//  a function to calculate the number of gallons of gasoline consumed.
#include <iostream>
#include <cstdlib>
using namespace std;
class Odometer
{
public:
void setFuelEfficiency(double newEfficiency);
void reset();
```

```
void logMiles(int additionalMiles);
double gasConsumed();
private:
int miles;
double fuel_efficiency;
};
// --------------------------------
// ----- ENTER YOUR CODE HERE -----
// --------------------------------
// ======================
// Odometer::setFuelEfficiency
// Sets the fuel efficiency in miles per gallon.
// ======================
void Odometer::setFuelEfficiency(double newEfficiency)
{
fuel_efficiency = newEfficiency;
}
// ======================
// Odometer::reset
// Resets the odometer reading
// ======================
void Odometer::reset()
{
miles = 0;
}
// ======================
// Odometer::addMiles
// Log additional miles to the odometer.
// ======================
void Odometer::logMiles(int additionalMiles)
{
```

```
miles += additionalMiles;
}
// ======================
// Odometer::gasConsumed
// Calculates the gallons of gas consumed on the trip.
// ======================
double Odometer::gasConsumed()
{
return (miles / fuel_efficiency);
}
// --------------------------------
// --------- END USER CODE --------
// --------------------------------
// ======================
//     main function
// ======================
int main()
{
  // Two test trips
  Odometer trip1, trip2;
void setFuelEfficiency(double newEfficiency);
void reset();
void logMiles(int additionalMiles);
double gasConsumed();
  trip1.reset();
  trip1.setFuelEfficiency(45);
  trip1.logMiles(100);
  cout << "For your fuel-efficient small car:" << endl;
  cout << "After 100 miles, " << trip1.gasConsumed() <<
  " gallons used." << endl;
  trip1.logMiles(50);
```

```
 cout << "After another 50 miles, " << trip1.gasConsumed() <<
" gallons used." << endl;
 cout << endl;
 trip2.reset();
 trip2.setFuelEfficiency(13);
 trip2.logMiles(100);
 cout << "For your gas guzzler:" << endl;
 cout << "After 100 miles, " << trip2.gasConsumed() <<
 " gallons used." << endl;
 trip2.logMiles(50);
 cout << "After another 50 miles, " << trip2.gasConsumed() <<
 " gallons used." << endl;
}
```

A3)

```
#include <iostream>
using namespace std;
class Money
{
public:
int getDollars();
int getCents();
void setDollars(int d);
void setCents(int c);
double getAmount();
private:
int dollars;
int cents;
};
int Money::getDollars()
{
return dollars;
```

```
}
int Money::getCents()
{
return cents;
}
void Money::setDollars(int d)
{
dollars = d;
}
void Money::setCents(int c)
{
cents = c;
}
double Money::getAmount()
{
return static_cast<double>(dollars) +
static_cast<double>(cents) / 100;
}
int main( )
{
Money m1, m2;
m1.setDollars(20);
m1.setCents(35);
m2.setDollars(0);
m2.setCents(98);
cout << m1.getDollars() << "." << m1.getCents() << endl;
cout << m1.getAmount() << endl;
cout << m2.getAmount() << endl;
cout << "Changing m1's dollars to 50" << endl;
m1.setDollars(50);
cout << m1.getAmount() << endl;
```

```
cout << "Enter a character to exit." << endl;
char wait;
cin >> wait;
return 0;
}
```

A4)

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;
void build_histogram(int letters[], string s);
void build_histogram(int letters[], string s)
{
for (int i = 0; i < 26; i++)
letters[i] = 0;
for (int i = 0; i < s.length(); i++)
{
char c = toupper(s[i]);
if (isalpha(c))
{
int index = static_cast<int>(c) -
static_cast<int>('A');
letters[index]++;
}
}
}
int main( )
{
string s1, s2;
int letter_histogram1[26], letter_histogram2[26];
```

```
cout << "Enter two strings." << endl;

getline(cin, s1);

getline(cin, s2);

build_histogram(letter_histogram1, s1);

build_histogram(letter_histogram2, s2);

// If histograms are identical they are anagrams

for (int i = 0; i<26; i++)

{

if (letter_histogram1[i] != letter_histogram2[i])

{

cout << "They are not anagrams." << endl;

char wait;

cin >> wait;

return 0;

}

}

cout << "They are anagrams!" << endl;

cout << "Enter a character to exit." << endl;

char wait;

cin >> wait;

return 0;

}
```

////////////////////////////// ikinci yol

```
// C++ program to check whether two strings are anagrams
// of each other
#include <bits/stdc++.h>
using namespace std;

/* function to check whether two strings are anagram of
each other */
bool areAnagram(string str1, string str2)
{
    // Get lengths of both strings
    int n1 = str1.length();
    int n2 = str2.length();

    // If length of both strings is not same, then they
    // cannot be anagram
```

```
    if (n1 != n2)
        return false;

    // Sort both the strings
    sort(str1.begin(), str1.end());
    sort(str2.begin(), str2.end());

    // Compare sorted strings
    for (int i = 0; i < n1; i++)
        if (str1[i] != str2[i])
            return false;

    return true;
}

// Driver code
int main()
{
    string str1 = "test";
    string str2 = "ttew";
    if (areAnagram(str1, str2))
        cout << "The two strings are anagram of each other";
    else
        cout << "The two strings are not anagram of each other";

    return 0;
}
```

A5)

```
#include <iostream>

#include <fstream>


using namespace std;


int main()

{

  fstream file; //object of fstream class


  //opening file "sample.txt" in out(write) mode

  file.open("sample.txt",ios::out);


  if(!file)

  {

     cout<<"Error in creating file!!!"<<endl;
```

```
   return 0;
}

cout<<"File created successfully."<<endl;
//write text into file
file<<"ABCD.";
//closing the file
file.close();

//again open file in read mode
file.open("sample.txt",ios::in);

if(!file)
{
   cout<<"Error in opening file!!!"<<endl;
   return 0;
}

//read untill end of file is not found.
char ch; //to read single character
cout<<"File content: ";

while(!file.eof())
{
   file>>ch; //read single character from file
   cout<<ch;
}

file.close(); //close file

return 0;
```

```
}
```