Data Structures TA Class #1

By Po-Chuan & Pei-Hsuan

104/10/12

Preface

Few things before we start

When doing your homework

1. Write in nice & clear format.
2. Please DO indent your code, and use monospaced fonts.
3. Answer all requirements one by one. Don’t skip any of them!
4. Do NOT hand in your homework in sheets unstapled.
5. Use the International Edition.

Problem 1-2

Fraction addition

Part A. Method for addition

- Purpose: add 2 fractions and return the result.
- Pre-condition: numerators & denominators are integers and denominators shall be non-zero.
- Post-condition: return the sum of the fractions (and both fractions are intact.)
- Parameter: a Fraction-class variable (or 2 integers.)
- Return value: the sum of these 2 fractions, which is also a Fraction.

Part A. grading policy

- 5 elements, each for 2 points.

Part B. C++ implementation

```
class Fraction
{
   private:
      int N, D;
      int gcd( int a, int b );
      void reduce();
   public:
      Fraction( int, int );
      Fraction add( const Fraction );
}; // Don't forget the terminating semicolon
```

```
/* GCD of numerator & denominator */
int Fraction::gcd( int a, int b )
{
   /* If a or b is 0, the GCD is 1 */
   if( !a || !b )
      return 1;
   /* If b | a, then b is GCD.
    * If not, return gcd( b, r ) */
   return a % b? gcd( b, a % b ) : b;
}
```

```
/* Reduce the fraction to the simple form */
void Fraction::reduce()
{
   int f = gcd( N, D );
   N /= f;
   D /= f;
}
```

```
/* Constructor */
Fraction::Fraction( int n = 1, int d = 1 ) :
N( n ), D( d )
{
   /* Denominator should not be zero */
   if( !D )
      throw "Denominator should not be 0\n";
   reduce();
}
```

```
/* Add 2 fractions */
Fraction Fraction::add( const Fraction addend )
{
    Fraction sum( N * addend.D + addend.N * D,
                  D * addend.D );
    sum.reduce();
    return sum; // Please do not return a Boolean
}
```

Part B. grading policy

- Comments… 1
- Other functions (GCD)… 1
- Syntax correctness… 1
- 通分… 2
- 加法… 2
- 約分… 1
- 回傳結果… 2

Problem 1-3

The appointment book

Part A. change purpose of a appointment

```
/* This is a pseudocode, not C++, function.
 * Appointment book is implemented in ADT
 * Please do not compile this code
 *
 * No new function is required. (But if you treat the following
 * as a new function, that's also ok.)
 * This function uses existing functions like isAppointment,
 * cancelAppointment, and makeAppointment.
 */
```

```
// Change the purpose of an appointment
updatePurpose( date: Date, time: Time, newPurpose: String )
{
    // Make sure the appointment exists
    if ( apptBook.isAppointment( date, time ) )
    {
        // The appointment does exist
        apptBook.cancelAppointment( date, time );
        if( apptBook.makeAppointment( date, time, newPurpose ) )
            write( “Purpose of the appointment updated." );
    }
    else // The appointment doesn't exist
        write( "No appointment at the given date & time." );
}
```

Part B. list all appointments in a day

```
/*
 * No new function is required. (But if you treat the following
 * as a new function, that's also ok.)
 * This function uses existing functions like isAppointment,
getAppointmentPurpose.
 */
```

Part B. list all appointments in a day

```
// Print all appointments for a given date
getDateAppointment( apptDate: Date )
{
    // Iterate through all possible times on the given day
    for( Time time: Date )
        // If the appointment exists
        if ( apptBook.isAppointment( date, time ) )
            write( apptBook.getAppointmentPurpose( date,
time ), " at ", date, time );
}
```

Grading policy

• Syntax correctness	1
• Pseudocode	3
• Function style	1
• Correctness	4
• Explanation	1

Problem 1-5

Remove elements in a bag & Javadoc comment style

Javadoc comment style

- Look at https://en.wikipedia.org/wiki/Javadoc
- Begins with /** and ends with */
- @param
  @return
- No blank lines between comment and code body
- Paragraphs in comment are separated by <p>

```
#include<bits/stdc++.h>
using namespace std;

/**
 * Class bag --- simulation of a bag of strings
 * Note that this problem asks you to comment in Javadoc style, so
 * please DO comment your functions, or you'll lose some marks.
 * <p>
 * Also, use `<p>' to separate paragraphs between descriptions when
 * needed, and there’s no blank lines between comment & function.
```

```
 *
 * @author     Po-Chuan
 * @version    1.5
 * @since      2015-10-12
 */
class bag
{
  private:
    vector<string> items;
```

```
public:
    /**
     * Remove items in bag and count how many items are removed.
     * <p>
     * This function iterates through the bag, searches for items and
     * removes those matched with the parameter. After iteration,
     * this function returns the number of items removed.
     *
     * @param   s   the item to be removed
     * @return      the number of the items that were removed
     */
    int remove( const string s )
```

```
int remove( const string s )
{
  /*
   * # of items removed
   */
  int rm = 0;
  /*
   * If the bag is empty, i is equal to s.end() and returns 0 immediately.
   * If none was found, no erase commands are invoked. A zero is returned.
   */
  for ( auto i = s.begin(); i != s.end(); ++i )
    if ( *i == s )              // Matched with target
      i = s.erase( i ), ++rm; // Erase and add 1 to the counter
  return rm;
}
};
```

Just some more…

- We use `vector` as a container here. You may use other containers.
- You can use `getFrequencyOf()` to count the number of objects.

Grading policy

• Count answer correctness	4
• Removal implementation	4
• Return value	2
• Comment (in Javadoc style)	3
• Function name	7

Problem 1-6

Union of 2 bags

```
#include<vector>
#include<algorithm>
using namespace std;

/* Class name: bag                                                  *
 * Purpose: simulate operations of a set                            *
 * Author: Po-Chuan                                                 *
 * Note: using template to achieve ADT style programming            *
 * Requirement: operator `<' for type T MUST be defined             */
```

```
template<typename T>
class bag
{
private:
    /* Use vector to hold contents */
    vector<T> contents;

public:
    /* Insert a new element into the vector */
    void insert( const T val )
    {
        contents.push_back( val );
    }
```

```
/* Do NOT use `bag&'!
 * Return a reference of a local variable will cause memory error */
bag Union( const bag& B )
{
    /* Insert all contents of the first set (*this) */
    bag merge = *this;

    /* Add all elements in B into the result */
    for( auto& i: B.contents )
        merge.insert( i );

    /* Assume the set is ordered. This part is optional */
    sort( merge.contents.begin(), merge.contents.end() );

    /* Return the merged bag */
    return merge;
}
};
```

Just some more…

- You can also use `getFrequencyOf()` to solve this problem.
- Get all frequencies of strings in bag A, and then get those in bag B.
- Push them all into the result.
- Remember to return a NEW bag. (A & B are intact.)

Grading policy

• Function name and comment	6
• Pseudocode	3
• Syntax correctness	2
• Return a new bag	3
• Union answer correctness	4
• Affect bag content	2

Problem 1-7

Intersection of 2 bags

```
#include<vector>
#include<iterator>
using namespace std;

template<typename T>
class bag
{
private:
    /* Use vector to hold contents */
    vector<T> contents;
    /* For internal use only, but it can also be public */
    void insert( const T& val )
    {
        contents.push_back( val );
    }
```

```
    /* Sort set contents */
    void sort()
    {
        sort( contents.begin(), contents.end() );
    }

public:

    /* Do NOT use bag&.
     * We'll modify B a bit, so pass by non-constant value */
    bag<T> intersection( bag B )
```

```
/* Specify the start & end point */

typename vector<T>::iterator
    first1 = A.contents.begin(),
    first2 = B.contents.begin(),
    last1 = A.contents.end(),
    last2 = B.contents.end();


/* Insert new elements from here */

insert_iterator< vector<T> > result( merge.contents,
    merge.contents.begin() );


/* Main intersection process */

/* You can also use
    set_intersection( first1, last1, first2, last2, result ); */
```

```
while ( first1 != last1 && first2 != last2 )
{
    if ( *first1 < *first2 )
        ++first1;
    else if ( *first2 < *first1 )
        ++first2;
    else
    {
        *result = *first1;
        ++result;
        ++first1;
        ++first2;
    }
}
```

```
        /* Return the merged bag */
        return merge;
    }
};
```

Just some more…

- Another way to solve it is `getFrequencyOf()`.
- For each kind of element in set A, count the frequency of that in set B.
- Insert the minimum of the 2 frequencies into the merged result.

Grading policy

• Function name and comment	4
• C++ code detail	3
• Syntax correctness	2
• ADT	2
• Return a new bag	3
• Intersection answer correctness	4
• Affect bag content	2

The end~

Hope you did a good job in this assignment.

Average score is 74.4.

By the TAs

104/10/12