CMPT 212 (2008-1) Assignment 2*
Due online Friday, February 22, 2008, 11pm

Ján Maňuch
jmanuch@sfu.ca

1 Programming Assignment

Write a C++ library implementing the class **`Set`** of integers (of type **`long`**). Your library will be split into two files:

1. the header file **`set.h`** containing the declaration of class **`Set`**;
2. the source code file **`set.cpp`** containing the implementation of class **`Set`**, that is definitions of all its class methods.

Note that none of your files should contain the **`main()`** function. The **`main()`** function will be provided in a *testing file* which will include (using **`#include "set.h"`** directive) your header file and will use the public class methods of the class **`Set`**. The executable file will be generated by linking together object file of **`set.cpp`** and the object file of the testing file.

Your program must compile under Visual Studio C++ 2005. See below the details how to compile your file. You will be penalized if some extra work is needed to compile your program on Visual Studio C++ 2005.

All your code must be located in two files called **`set.h`** and **`set.cpp`**, which will be the only files submitted for the assignment. You can create your own testing file for testing your library, but do not send your testing file in your submission!

***** Important *****
You are not allowed to use any library beside **`iostream`**. That is:

- Your header file **`set.h`** should include only **`iostream`** file.
- Your source code file **`set.cpp`** should include only **`set.h`** file.

Information about evaluating and grading the assignment can be found at the course website.

If you have any questions about the assignment, do not hesitate to send me an email. I might revise the text of the assignment based on your questions, if I find out that some parts are difficult to understand or insufficiently specified.

2 Problem Specification

2.1 Interface

The class **`Set`** should implement the following *interface* described in the **`public`** section of the class.

```
class Set {
 private:
  // implementation details:
  // ...
 public:
```

Revision : 1.0 (Friday 8th February, 2008,16:17); Details in this document may change before the date of the submission of the assignment. Please make sure you have the latest version.

```
  // public interface:
  Set(); // the default constructor
         //  - initializes the set to empty set
  Set(long a, long b, long c=1); // initializes the set to the set
    // containing the elements a, a+c, a+2*c, a+3*c, ...
    // which are smaller or equal to b
  Set(long list[],long size); // initializes the set to elements
    // contained in the array list; the size should be size of the array
    // (note: you cannot assume that elements in list are all distinct)
  Set(const Set &s); // the copy constructor
  ~Set(); // destructor

  // if the following methods return a reference to Set,
  // it should be reference to itself (which allows chaining of the operations)
  Set & operator=(const Set &s); // the assignment operator
  operator long() const; // return the number of elements
  bool operator()(long x) const; // return true if x is an element of the set
  bool operator==(const Set &s) const;
    // return true if s contains the same elements
  bool operator<=(const Set &s) const;
    // return true if this object is subset of s
  bool operator>=(const Set &s) const;
    // return true if s is subset of this object

  Set & operator<<(long x); // add x to the set
  Set & operator>>(long x); // remove x from the set
    // (if x is not in the set, do nothing)
  Set operator+(const Set &s) const; // return the union with the set s
  Set & operator+=(const Set &s); // add all elements of the set s
  Set operator*(const Set &s) const; // return the intersection with the set s
  Set & operator*=(const Set &s);
    // keep only those elements which are also in s
  Set operator-(const Set &s) const; // return the set difference with the set s
  Set & operator-=(const Set &s); // remove those elements which are also in s
  friend ostream & operator<<(ostream &os, const Set &s);
    // print the content of the set s
};
```

2.2 Implementation

- The **`private`** section of the class should contain the implementation details:
  - *data members* representing the content of the set in memory — this is completely up to you, you can represent the set as an array containing the actual elements (in sorted or unsorted manner), or a linked list of all elements (in sorted or unsorted manner), etc.;
  - *auxiliary functions* which are not part of the interface but you need them to handle the data [you might not need any of such functions, it depends on your implementation].
- The **`public`** section of the class should contain only the interface described above. *You are not allowed to add or remove any functions to/from the interface, neither to modify the prototypes of these functions!* The **`public`** section should not contain any data members.
- The implementation of all member functions (auxiliary ones and the interface functions) should be contained in **`set.cpp`** file, not in **`set.h`** file. Do not add function bodies to the class declaration in **`set.h`** file.

- If your implementation dynamically allocates memory (using `new` operator), your destructor is supposed to free (using `delete` operator) **all** allocated memory when to object seize to exist. You might be penalized if your program is causing memory leaks.
- If the object of class `Set` is sent to `ostream`, it should be printed in the following format:

  `{comma-separated-list-of-elements-in-sorted-order}`

  For instance, the set containing three elements -2, 5 and 7 should be printed as `{-2,5,7}`. Note the output should not contain any spaces. The empty set should be printed as `{}`. Please, make sure that you print elements correctly ordered and that you don't print anything else than required (no new line characters, no spaces). It will simplify the evaluation process. *You will get penalized if your program prints the sets in a different way.*
- No other function than `Set::operator<<()` is supposed to print anything. Again, any other output from the program than required will make the evaluation process difficult and you will be penalized for that.

You don't have to type the interface again, here is the start-up header file: set.h.

2.3 Remarks

- When creating your own test files, please, beware of the following:

  ```
  Set A;
  A << 10;
  ```

  gives you a compiler error in the second line: ambiguous call. The reason is that `Set::operator<<` is expecting `long`, not `int` (10), hence to call the operator a conversion `int` to `long` has to be performed. At the same time `Set` can be converted to `long` (the number of elements) and the built-in `<<` operator is defined on the combination of types `long` and `int`. Hence, the compiler cannot choose which conversion to perform.
- Try to make your solution efficient enough. It's impossible to spend more than 60 seconds per program per test file. The programs which will take longer than that will not be fully marked: only output generated during first 60 seconds (on a 3.0Ghz P4) will be marked.

2.4 Testing

This section contains an example of the testing file. This testing file together with several other testing files will be used to evaluate your submission. Hence, you should certainly make sure that your library works correctly with this testing file. You can also make your own testing files to ensure that your library works correctly with other testing files.

Here is the testing file #1 (a2_test1.cpp):

```
#include <iostream>
#include "set.h"

using namespace std;

int main()
{
  Set A; // empty set
  cout << "A: " << A << endl;

  A <<20L <<37L << 305L; // add elements 20, 37, 305
  A <<10345L <<-199L;   // add elements 10345, -199
  cout << "A: "<< A << endl;

  cout << boolalpha;
```

```
    // force cout to print 'true' and 'false' for bool values
  cout << "305 is a member of A: " << A(305) << endl;

  A >>304L >>305L;  // remove 304 and 305 from A
  cout << "A: "<< A << endl;
  cout << "305 is a member of A: " << A(305) << endl;

  Set B(1,100); // set {1,2,3,...,100}
  long Bsize=B;
  cout << "Number of elements of B: " << Bsize << endl;

  Set C=A; // a copy of A
  cout << "C: " << C << endl;

  C *= B; // C=C intersection B
  cout << "C: " << C << endl;
  cout << "A: " << A << endl; // A was not modified
  cout << "C = A: " << (C==A) << endl;
  cout << "C subset of A: " << (C<=A) << endl;

  cout << "C U Set(10,32,7): " << C+Set(10,32,7) << endl;

  A -= B; // A=A-B; set difference
  cout << "A: " << A << endl;

  return 0;
}
```

This testing file should produce the following output:

```
A: {}
A: {-199,20,37,305,10345}
305 is a member of A: true
A: {-199,20,37,10345}
305 is a member of A: false
Number of elements of B: 100
C: {-199,20,37,10345}
C: {20,37}
A: {-199,20,37,10345}
C = A: false
C subset of A: true
C U Set(10,32,7): {10,17,20,24,31,37}
A: {-199,10345}
```

2.5 Compilation

To compile your library together with a testing file (either **`a2_test1.cpp`** or your own testing file) under Visual Studio C++ 2005, create a new project and add two files: **`set.cpp`** and the testing file (e.g., **`a2_test1.cpp`**). Do not add the header file **`set.h`** into project, the header file will be included automatically when preprocessor sees the directive **`#include "set.h"`**.

To compile the files on GCC compiler, you can either compile them together:

```
g++ a2_test1.cpp set.cpp
```

or you can first explicitly create the object files and then link them together:

```
g++ -c set.cpp
g++ -c a2_test1.cpp
g++ a2_test1.o set.o
```

Note that if you do not specify the output filename,the executable file is called **a** or **a.out**.

3 Programming Contest

Assignment 2 will have an associated programming contest. Only assignments that are considered to have adequate implementations, as outlined in Section 2 will be entered into the contest.

All qualifying entries will be compiled and tested on Linux, using the GCC compiler, similar with the one installed on the MTF Linux lab machines (e.g. **tomato.csil.sfu.ca**). The winners will be entries that result in the *fastest running time on testing files*. Your program will be tested on two testing files: one performing a lot of operations keeping the size of the sets small, while the other performing a few operations with sets containing many elements (approx. 10,000). The score assigned to your program will be the sum of the running times on those 2 testing files.

Up to three entries will be selected. In case of ties that result in more than three top entries of equal size, none of the tied entries will be considered.

The winners will be publicly acknowledged on the course website and during lecture. The source code of winning assignments will be uploaded to and linked from the course web site. Students who are not comfortable with these conditions should indicate that through email at time of submission of their assignment and will not be considered for the contest.