1 Instruction

When you are finished submit all your work through the MyClasses page for this class. Create a directory called Homework10, put each programming exercise into its own subdirectory of this directory, zip the entire Homework10 directory up into the file Homework10.zip, and then submit this zip file to Homework #10.

Make sure that you:

- Follow the coding and documentation standards for the course as published in the MyClasses page for the class.
- Check the contents of the zip file before uploading it. Make sure all the files are included.
- Make sure that the file was submitted correctly to MyCLasses.

All non-templated class structures are to have their own guarded specification file (.h) and implementation file (.cpp) that has the same name as the class. All templated class structures are to be guarded and written entirely in their (.h) file. No inline coding in the class specification. In addition you must create a make file that compiles and links the project on a Linux computer with a Debian or Debian branch flavor.

2 Exercise #1

Write a program that will ask the user to input the number of dice to roll, the number of sides of the dice (assume all die are the same), and the number of rolls. Have the program roll all the dice that many times and store the *counts* of the total roll in a map structure. Then have the program print out the contents of the map structure. Example runs are below.

```
Input number of dice to roll: 2
Input number of sides for each die: 6
Input number of rolls of dice: 10
Results
4 1
6 3
7 3
8 1
9 1
10 1
```

```
Input number of dice to roll: 3
Input number of sides for each die: 6
Input number of rolls of dice: 1000000
Results
3 4671
4 13737
5 27774
6 46572
7 69237
```

```
8 97385
9 115515
10 125185
11 124960
12 115617
13 97671
14 69185
15 46294
16 27708
17 13875
18 4614
```

```
Input number of dice to roll: 3
Input number of sides for each die: 4
Input number of rolls of dice: 1000000
Results
3 15666
4 47217
5 93968
6 156358
7 187433
8 187339
9 156272
10 93766
11 46349
12 15632
```

3 Exercise #2

Revise the above program to store all the rolls in a multiset and then use the count function to display the same results. Make sure that each roll total appears only one time. Example runs are below.

```
Input number of dice to roll: 3
Input number of sides for each die: 6
Input number of rolls of dice: 10
Results
4 1
5 1
7 1
8 2
9 1
10 1
11 1
15 1
17 1
```

```
Input number of dice to roll: 2
Input number of sides for each die: 6
Input number of rolls of dice: 1000000
Results
2 27896
3 55596
4 83531
5 110661
6 139481
7 166358
```

```
8 138748
9 111262
10 83371
11 55218
12 27878

Input number of dice to roll: 2
Input number of sides for each die: 4
Input number of rolls of dice: 1000000
Results
2 61954
3 124703
4 187578
5 250120
6 187825
7 125267
8 62553
```

4 Exercise #3

Create a class called MathSet that inherits off of the STL set class. The MathSet class will be an extension of the set class that does some of the operations commonly seen in set theory and discrete mathematics. The MathSet is to be templated but it does not need to store any data, the set parent class will take care of all that. Overload the following operators,

- + : This is to return a set that is the union of the two operand sets. So `A + B` is in mathematical notation $A \cup B$.
- − : This is to return a set that is the set difference of the two operand sets. So `A - B` is the set of all elements of A that are not in B,
- ⋆ : This is to return a set that is the intersection of the two operand sets. So `A * B` is in mathematical notation $A \cap B$.
- ˆ : This is to return a set that is the symmetric difference of the two operand sets. So `A ^ B` is the set of all elements which are in either of the sets, but not in their intersection.
- << : stream out operator that prints out the set in standard mathematical notation, as in the examples below.
- As you would expect, the overloaded = and constructors pass right through to the set class, with a couple exceptions. The constructor with the initializer list does not pass through automatically, so we will create one. This, in turn, requires an overloaded default constructor. Both of these simply pass the construction to the base set class as we have done many times before with child classes. Without this initializer constructor the main program code of

  ```
  MathSet<int> A = {1, 2, 3, 4, 5, 6};
  ```

  would not work. There are other things we could do to integrate this further but for now this is sufficient.

  - Create a default constructor that simply calls the set default constructor.
  - Create a constructor that brings in an initializer list and invokes the set constructor with that list. The start of the specification entry is
    `MathSet<T>(initializer_list<T> L)`
    finish it.

4.1 Extra Credit

For some extra credit, write a function that creates the power set of the set. The power set of a set A is denoted as $\mathcal{P}(A)$, and is defined as the set of all subsets of A. So for example, if $A = \{1, 2, 3\}$, then $\mathcal{P}(A) = \{\{\}, \{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 3\}, \{2\}, \{2, 3\}, \{3\}\}$. The power set can be defined recursively as,

- If $S = \{\}$, then $\mathcal{P}(S) = \{\{\}\}$.
- If $S \neq \{\}$, let $e \in S$ and $T = S - \{e\}$, then $\mathcal{P}(S) = \mathcal{P}(T) \cup \{t \cup \{e\} : t \in \mathcal{P}(T)\}$.

Use this definition to code the power set function.

4.2 Program Run

The following test program produces the output below. Note that if you did not do the extra credit you should comment out the portion that tests the power set function.

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

using namespace std;

int main() {
  srand(time(0));

  MathSet<int> A = {1, 2, 3, 4, 5, 6};
  cout << A << endl;

  MathSet<int> B;
  B.insert(17);
  cout << B << endl;

  MathSet<int> C;
  cout << C << endl;

  B.insert({
      4,
      5,
      6,
      7,
  });
  cout << B << endl;

  cout << A + B << endl;
  cout << A - B << endl;
  cout << B - A << endl;
  cout << A * B << endl;
  cout << (A ^ B) << endl;

  B = A;
  cout << A << endl;
  cout << B << endl;

  B = {4, 3, 6, 7};

  A.insert({10, 11, 12});
  cout << A << endl;
  cout << B << endl;

  MathSet<int> E = A;
  cout << A << endl;
  cout << E << endl;

  MathSet<int> F(A);
  cout << A << endl;
  cout << F << endl;

  cout << &A << endl;
  cout << &E << endl;
  cout << &F << endl;

  MathSet<int>::iterator it = A.begin();
  while (it != A.end())
    cout << *it++ << " ";
  cout << endl;

  it = A.find(6);
  if (it != A.end())
    A.erase(it);
  cout << A << endl;

  A.erase(10);
  cout << A << endl;

  it = A.find(3);
  auto it2 = A.find(11);
  A.erase(it, it2);
  cout << A << endl;

  cout << "------------- Power Set Test Code" << endl;

  A.clear();
  A.insert({1, 2, 3});
  cout << A << endl;
  cout << A.PowerSet() << endl;
  A.insert(4);
  cout << A.PowerSet() << endl;

  MathSet<char> S = {'A', 'B', 'C'};
  cout << S.PowerSet() << endl;

  S.insert('D');
  cout << S.PowerSet() << endl;

  auto D = S.PowerSet();
  cout << D << endl;

  return 0;
}
```

Output:

```
{1, 2, 3, 4, 5, 6}
{17}
{}
{4, 5, 6, 7, 17}
{1, 2, 3, 4, 5, 6, 7, 17}
{1, 2, 3}
{7, 17}
{4, 5, 6}
{1, 2, 3, 7, 17}
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6, 10, 11, 12}
{3, 4, 6, 7}
{1, 2, 3, 4, 5, 6, 10, 11, 12}
{1, 2, 3, 4, 5, 6, 10, 11, 12}
{1, 2, 3, 4, 5, 6, 10, 11, 12}
{1, 2, 3, 4, 5, 6, 10, 11, 12}
0x7ffc63e12a20
0x7ffc63e12ae0
0x7ffc63e12b10
1 2 3 4 5 6 10 11 12
{1, 2, 3, 4, 5, 10, 11, 12}
{1, 2, 3, 4, 5, 11, 12}
{1, 2, 11, 12}
------------- Power Set Test Code
{1, 2, 3}
{{}, {1}, {1, 2}, {1, 2, 3}, {1, 3}, {2}, {2, 3}, {3}}
{{}, {1}, {1, 2}, {1, 2, 3}, {1, 2, 3, 4}, {1, 2, 4}, {1, 3}, {1, 3, 4}, {1, 4}, {2},
     {2, 3}, {2, 3, 4}, {2, 4}, {3}, {3, 4}, {4}}
{{}, {A}, {A, B}, {A, B, C}, {A, C}, {B}, {B, C}, {C}}
{{}, {A}, {A, B}, {A, B, C}, {A, B, C, D}, {A, B, D}, {A, C}, {A, C, D}, {A, D}, {B},
     {B, C}, {B, C, D}, {B, D}, {C}, {C, D}, {D}}
{{}, {A}, {A, B}, {A, B, C}, {A, B, C, D}, {A, B, D}, {A, C}, {A, C, D}, {A, D}, {B},
     {B, C}, {B, C, D}, {B, D}, {C}, {C, D}, {D}}
```