

# 1 Introduction

Each exercise should be its own separate project.

**Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.**

When you are ready to submit your work create a folder called Homework12 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Homework12 folder up into a single zip file and submit it.

## 2 Exercises

1. In the last homework you created a class called `IntList` that stored a size, capacity, and a pointer to an integer. Here you will add in some operators.

The class specification is given below. The first set of functions are as in the last homework and the new functions and operators are below those, descriptions of the new functions are below.

```
class IntList {
private:
    int size = 0;
    int capacity = 0;
    int *list = nullptr;

public:
    IntList();
    ~IntList();

    void set(int, int);
    int get(int);
    int get_size();
    int get_capacity();
    void push_back(int);
    void push_front(int);
    int pop_back();
    int pop_front();
    void concat(const IntList&);
    void increase_capacity();
    void sort();
    void print();

    // New Functions
    void clear();
    IntList(const IntList&);
    const IntList operator=(const IntList&);

    int& operator[] (const int&);

    IntList operator+(const IntList&);
    void operator +=(const IntList&);

    bool operator ==(const IntList&);
```

```
bool operator !=(const IntList&);  
  
friend ostream& operator <<(ostream&, const IntList&);  
};
```

- `clear`: Will clear the contents of the integer list.
- `IntList(const IntList&)`: Is the copy constructor.
- `operator=`: Overloaded assignment statement.
- `operator[]`: Overloaded array accessor. If the index is out of bounds then the program should display an error and halt.
- `operator+`: Will concatenate the two lists and return the result. Neither of the two lists in the expression are to be altered.
- `operator+=`: Will concatenate the two lists and store the result in the left list of the expression.
- `operator==`: Overloaded equal operator should return true if both lists are identical and false if they differ.
- `operator!=`: Overloaded not equal operator should return true if the lists differ and false if they are the same.
- `operator <<`: Output stream operator should output the list elements on one line with a space between each element.

Once these functions are written the following main program will produce the following output.

```
#include <iostream>  
#include "IntList.h"  
  
using namespace std;  
  
void doNothing(IntList l) {  
    cout << l << endl;  
    l[4] = 1;  
    l[0] = 1029;  
}  
  
int main() {  
    IntList list1, list2, list3;  
  
    list1.push_back(15);  
    list1.push_back(-4);  
    list1.push_back(3);  
    list1.push_back(7);  
    list1.print();  
    cout << list1.get_size() << " " << list1.get_capacity() << endl;  
  
    list1.push_back(17);  
    list1.print();  
    cout << list1.get_size() << " " << list1.get_capacity() << endl;  
  
    for (int i = 1; i <= 10; i++)  
        list2.push_back(i);
```

```
list2.print();
cout << list2.get_size() << " " << list2.get_capacity() << endl;

cout << list2.get(5) << endl;
cout << list2.get(15) << endl;

list2.set(5, 1234);
list2.set(15, -5);
list2.print();

list2.push_front(3);
list2.print();
cout << list2.get_size() << " " << list2.get_capacity() << endl;

cout << list2.pop_front() << endl;
cout << list2.pop_front() << endl;
cout << list2.pop_back() << endl;

list2.print();
cout << list2.get_size() << " " << list2.get_capacity() << endl;

cout << endl;
list1.print();
list2.print();

cout << endl;
list1.concat(list2);
list1.print();
cout << list1.get_size() << " " << list1.get_capacity() << endl;

list2.print();
cout << list2.get_size() << " " << list2.get_capacity() << endl;

cout << endl;
list1.push_back(101);
list1.print();
cout << list1.get_size() << " " << list1.get_capacity() << endl;

list1.sort();
list1.print();
cout << list1.get_size() << " " << list1.get_capacity() << endl;

// New function testing

cout << "New function testing ~~~~~~" << endl;

list1.clear();
list1.push_back(5);

list1.print();

for (int i = 1; i <= 10; i++)
    list1.push_back(3 * i + 1);

list1.print();

list2 = list1;

cout << endl;
list1.print();
list2.print();

list1.set(3, -12);
list1.set(7, 12345);
```

```

    cout << endl;
    list1.print();
    list2.print();

    cout << list1[3] << endl;
    cout << list2[3] << endl;
    cout << list2[4] << endl;
    cout << list2[7] << endl;

    // cout << list2[100] << endl; // Error causes program exit.
    // cout << list3[1] << endl; // Error causes program exit.

    cout << list2 << endl;

    list2[4] = -16;
    list2[1] = -2;
    cout << list2 << endl;

    list3 = list1 + list2;

    cout << endl;
    cout << list1 << endl;
    cout << list2 << endl;
    cout << list3 << endl;

    if (list1 == list2)
        cout << "list1 == list2" << endl;

    if (list1 != list2)
        cout << "list1 != list2" << endl;

    if (list1 == list1)
        cout << "list1 == list1" << endl;

    doNothing(list2);
    cout << list2 << endl;

    list2 += list3;
    cout << list2 << endl;

    return 0;
}

```

Output:

```

15 -4 3 7
4 4
15 -4 3 7 17
5 8
1 2 3 4 5 6 7 8 9 10
10 16
6
List bounds error, returning 0.
0
List bounds error.
1 2 3 4 5 1234 7 8 9 10
3 1 2 3 4 5 1234 7 8 9 10
11 16
3
1
10
2 3 4 5 1234 7 8 9
8 16

```

```

15 -4 3 7 17
2 3 4 5 1234 7 8 9

15 -4 3 7 17 2 3 4 5 1234 7 8 9
13 13
2 3 4 5 1234 7 8 9
8 16

15 -4 3 7 17 2 3 4 5 1234 7 8 9 101
14 26
-4 2 3 3 4 5 7 7 8 9 15 17 101 1234
14 26
New function testing ~~~~~
5
5 4 7 10 13 16 19 22 25 28 31

5 4 7 10 13 16 19 22 25 28 31
5 4 7 10 13 16 19 22 25 28 31

5 4 7 -12 13 16 19 12345 25 28 31
5 4 7 10 13 16 19 22 25 28 31
-12
10
13
22
5 4 7 10 13 16 19 22 25 28 31
5 -2 7 10 -16 16 19 22 25 28 31

5 4 7 -12 13 16 19 12345 25 28 31
5 -2 7 10 -16 16 19 22 25 28 31
5 4 7 -12 13 16 19 12345 25 28 31 5 -2 7 10 -16 16 19 22 25 28 31
list1 != list2
list1 == list1
5 -2 7 10 -16 16 19 22 25 28 31
5 -2 7 10 -16 16 19 22 25 28 31
5 -2 7 10 -16 16 19 22 25 28 31 5 4 7 -12 13 16 19 12345 25 28 31 5 -2 7 10 -16 16 19 22 25 28 31

```

- Now we are going to take the integer list from above and create a set structure. A set is a collection of objects without repetition. Here we will create a set of integers. The STL contains a set structure and although it is a bit different than a mathematical set you can do some of these manipulations with this set structure. For this exercise you may not use any STL functionality including vectors. The specification to the class is below and descriptions follow.

```

class IntSet {
private:
    int size = 0;
    int capacity = 0;
    int *list = nullptr;

    void removeDuplicates();
    void increase_capacity();
    void push_back(int);

public:
    IntSet();
    IntSet(const IntSet&);
    ~IntSet();

    int get_size();
    void add(const IntSet&);
    void add(int);
    bool in(int);
    bool isEmpty();

```

```

void sort();
void print();

void clear();
const IntSet operator=(const IntSet&);

IntSet operator+(const IntSet&);
IntSet operator-(const IntSet&);
IntSet operator*(const IntSet&);

bool operator==(const IntSet&);
bool operator!=(const IntSet&);
bool operator>(const IntSet&);
bool operator<(const IntSet&);

friend ostream& operator <<(ostream&, const IntSet&);
};

```

The functions `increase_capacity`, `push_back`, `get_size`, `add`, `in`, and `sort` as well as the constructors and destructors are the same as in the `IntList` class and should require little to no alteration. The `isEmpty` function will return true if the set is empty and false otherwise. The `print` function will display the contents of the list with a comma between the entries and enclosed in `{}`. `Clear` will remove the contents of the set. The `removeDuplicates` function will remove any duplicate entries in the set. The overloaded operators are as follows, the `=` is assignment of one set to another. The `+` is the union and `*` is the intersection. The `-` is set difference, set difference is defined as  $A - B$  means the set of all elements in the set  $A$  that are not in the set  $B$ . The logical operators `==` is set equality, that is the sets contain the same elements, `!=` being the opposite. The `>` would be the superset and `<` would be the subset. So if  $A < B$  was true then  $A$  is a subset of  $B$  and if  $A > B$  is true then  $B$  is a subset of  $A$ . Finally the `<<` output stream operator displays the set as the `print` function does. Once the class is written the following testing program will produce the following output.

```

#include <iostream>
#include "IntSet.h"

using namespace std;

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

    IntSet set1, set2, set3;

    cout << set1 << endl;

    for (int i = 1; i <= 10; i++)
        set1.add(i);

    cout << set1 << endl;

    for (int i = 5; i <= 15; i++)
        set1.add(i);

    cout << set1 << endl;

    set1.add(-5);
    cout << set1 << endl;
}

```

```
set1.sort();
cout << set1 << endl;

set1.clear();
cout << set1 << endl;

if (set1.isEmpty())
    cout << "set1 is empty." << endl;

cout << endl;

for (int i = 1; i <= 10; i++)
    set1.add(i);

for (int i = 5; i <= 15; i++)
    set2.add(i);

for (int i = -15; i <= -7; i++)
    set2.add(i);

cout << set1 << endl;
cout << set2 << endl;

set3 = set1 + set2;
cout << set3 << endl;

set3 = set1 * set2;
cout << set3 << endl;

set3 = set1 - set2;
cout << set3 << endl;

set3 = set2 - set1;
cout << set3 << endl;

set3 = set2 - set2;
cout << set3 << endl;

cout << endl;
if (set2.in(8))
    cout << "8 is in set 2." << endl;

if (set2.in(100))
    cout << "100 is in set 2." << endl;

cout << endl;
set1.clear();

for (int i = 1; i <= 10; i++)
    set1.add(i);

set2 = set1;

for (int i = 100; i <= 110; i++)
    set2.add(i);

cout << set1 << endl;
cout << set2 << endl;

cout << (set1 < set2) << endl;
cout << (set1 > set2) << endl;
cout << (set1 == set2) << endl;
cout << (set1 != set2) << endl;
cout << (set2 == set2) << endl;
```

```

    set1.clear();
    set2.clear();

    cout << endl;

    for (int i = 0; i < 10; i++)
        set1.add(rand() % 100 + 1);

    for (int i = 0; i < 10; i++)
        set2.add(rand() % 100 + 1);

    cout << set1 << endl;
    cout << set2 << endl;

    set1.add(set2);

    cout << endl;
    cout << set1 << endl;
    cout << set2 << endl;

    return 0;
}

```

Program output:

```

{}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, -5}
{-5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
{}
set1 is empty.

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, -15, -14, -13, -12, -11, -10, -9, -8, -7}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, -15, -14, -13, -12, -11, -10, -9, -8, -7}
{5, 6, 7, 8, 9, 10}
{1, 2, 3, 4}
{11, 12, 13, 14, 15, -15, -14, -13, -12, -11, -10, -9, -8, -7}
{}

8 is in set 2.

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110}
1
0
0
1
1

{39, 52, 66, 25, 11, 88, 62, 97, 34}
{67, 52, 31, 40, 74, 14, 27, 100, 63, 64}

{39, 52, 66, 25, 11, 88, 62, 97, 34, 67, 31, 40, 74, 14, 27, 100, 63, 64}
{67, 52, 31, 40, 74, 14, 27, 100, 63, 64}

```

- For the final program use the set class you created in the previous exercise to create the following guessing game. Create a set of 100 integers between the values of 1 and 1000 randomly. Have the user guess a number in the set. If they guess a correct number in the set they win the game. If they do not guess correctly tell them the number of numbers in the set that are within 10 of the number they guessed. That is, if they



guess  $g$  and it is not in the set tell them the number of numbers in the set that are in the range  $[g - 10, g + 10]$ . The player will get 10 tries to guess a number in the set. If they do not guess any number in the set then they lose the game. Two runs of the game are below.

```
Guess a number (try 1): 100
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 2): 110
You did not guess a number in the set.
There are 1 numbers in the set within 10 of your guess.
Guess a number (try 3): 90
You did not guess a number in the set.
There are 3 numbers in the set within 10 of your guess.
Guess a number (try 4): 95
You did not guess a number in the set.
There are 1 numbers in the set within 10 of your guess.
Guess a number (try 5): 85
You did not guess a number in the set.
There are 4 numbers in the set within 10 of your guess.
Guess a number (try 6): 86
You did not guess a number in the set.
There are 4 numbers in the set within 10 of your guess.
Guess a number (try 7): 87
You did not guess a number in the set.
There are 4 numbers in the set within 10 of your guess.
Guess a number (try 8): 88
You did not guess a number in the set.
There are 4 numbers in the set within 10 of your guess.
Guess a number (try 9): 89
You did not guess a number in the set.
There are 3 numbers in the set within 10 of your guess.
Guess a number (try 10): 84
Game Over: You did not guess a number in the set.
```

```
Guess a number (try 1): 10
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 2): 20
You did not guess a number in the set.
There are 0 numbers in the set within 10 of your guess.
Guess a number (try 3): 5
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 4): 7
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 5): 8
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 6): 9
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 7): 4
You did not guess a number in the set.
There are 2 numbers in the set within 10 of your guess.
Guess a number (try 8): 3
You guessed a number in the set.
```