EEEC10008(515169) S23: Object-Oriented Programming
Standard Libraries on Containers and Algorithms

What you will learn from Lab 11

In this laboratory, you will learn how to use STL containers and generic algorithms provided by standard library.

TASK 11-1 VECTOR

✓ A container is an object whose main purpose is to hold other objects. A `vector` contains an array of `n` objects indexed from `0` to `n-1`.

```
// lab11-1.cpp
#include <iostream>
#include <vector>
using std::cout;
using std::endl;
using std::vector;

int main()
{
   int n = 10;
   vector<int> vec1(n);            // allocate a vector with 10 elements

   for (int i = 0; i < vec1.size(); i++)
      vec1[i] = i * i;             // use subscripting to access elements

   for (int i = 0; i < vec1.size(); i++)
      cout << vec1[i] << " ";
   cout << endl;

   vector<int> vec2;               // allocate an empty vector
   for (int i = 0; i < n; i++)
      vec2.push_back(i * 2);       // use push_back() to add elements

   vector<int>::const_iterator iter;
   for (iter = vec2.begin(); iter != vec2.end(); iter++)
      cout << iter << " ";         // use iterator to traverse container
   cout << endl;

   return 0;
}
```

- Please fix the compiling error here.
- Note that, `vec1[i]` and `vec1.at(i)` are similar to access elements in vector. However, `vec1.at(i)` provides range checking but `vec1[i]` does not.

- ✓ A vector of class objects can be created if the class has a default constructor.

```
// lab11-2.cpp
#include <iostream>
#include <vector>
using std::cout;
using std::endl;
using std::vector;
using std::ostream;

class Point2D
{
private:
    int x;
    int y;
public:
    Point2D(): x(0), y(0){}
    Point2D(int a, int b): x(a), y(b){}
    friend ostream &operator << (ostream &out, const Point2D &p)
    {
        out << "(" << p.x << "," << p.y << ")";
        return out;
    }
};

int main()
{
    int n = 10;
    vector<Point2D> vec(n);          // call Point2D()

    for (int i = 0; i < vec.size(); i++)
        vec[i] = Point2D(i*2, i*3);   // call Point2D(int a, int b)

    for (int i = 0; i < vec.size(); i++)
        cout << vec[i] << " ";
    cout << endl;

    return 0;
}
```

- ✓ Here demonstrate more operations supported by `vector`.

```
// lab11-3.cpp
#include <iostream>
#include <vector>

using std::cout;
using std::endl;
using std::vector;

int main()
```

```
{
    int n = 5;
    vector<int> vec(n,-1);                   // vec = {-1,-1,-1,-1,-1}
    vector<int> u(3);
    for (int i = 0; i < 3; i++) u[i] = i;    // u = {0,1,2}

    vec.insert(vec.begin()+2, u.begin(), u.end());
                                       // vec = {-1,-1, 0,1,2,-1,-1,-1}
    vec.insert(vec.begin()+1, 10);     // vec = {-1,10,-1,0,1, 2,-1,-1,-1}
    vec.pop_back();                    // vec = {-1,10,-1,0,1, 2,-1,-1}
    vec.erase(vec.begin()+3);          // vec = {-1,10,-1,1,2,-1,-1}
    vec.clear();                        // vec = {}

    for (int i = 0; i < vec.size(); i++)
        cout << vec[i] << " ";
    cout << endl;

    return 0;
}
```

- The functions `begin()` and `end()` return iterators to the first element and one-past-the-last element, respectively. It denotes the interval `[begin,end)`.
- `vec.insert(p,x)` is used to add element `x` at position `p` and `vec.insert(p, first, last)` can insert a sequence `[first,last)` to position `p`.
- `vec.erase(p)` remove the element at position `p`
- `vec.clear()` remove all elements.

✓ In `<algorithm>`, `sort()` is defined to sort the elements in increasing order. `reverse()` can reverse the elements in container and `find()` is used to find the specific element.

```
// lab11-4.cpp
#include <iostream>
#include <algorithm>
#include <vector>

using std::cout;
using std::endl;
using std::vector;
using std::ostream;

int main()
{
    int n = 10;
    vector<int> vec(n);                    // vec here is just an example.

    for (int i = 0; i < vec.size(); i++) // vec = {3,6,7,5,3,5,6,2,9,1}
        vec[i] = rand()%n;
    sort(vec.begin(), vec.end());         // vec = {1,2,3,3,5,5,6,6,7,9}
    reverse(vec.begin(), vec.end());      // vec = {9,7,6,6,5,5,3,3,2,1}

    for (int i = 0; i < vec.size(); i++)
       cout << vec[i] << " ";
    cout << endl;

    vector<int>::iterator iter = find(vec.begin(),vec.end(),8);
    if (iter != vec.end())
      cout << "8 is in the vector." << endl;
    else
      cout << "8 is not in the vector." << endl;

    return 0;
}
```

- The function `sort()` sort elements of the vector in increasing order based on a less-than operation < by default.

✓ In `<algorithm>`, `sort()`can sort the elements according to a compared function defined by user.

```
// lab11-5.cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

class A{
    public:
    int a,b;
};

bool compare(const A &c1, const A &c2){
    if(c1.a < c2.a) return true;
    else if(c1.a == c2.a && c1.b < c2.b) return true;
    else return false;
}

int main()
{
    vector<A> vec(3);
    vec[0].a = 1;  vec[0].b = 2;
    vec[1].a = 3;  vec[1].b = 2;
    vec[2].a = 3;  vec[2].b = 3;
    sort(vec.begin(), vec.end(), compare);

    for (int i = 0; i < vec.size(); i++)
       cout << vec[i].a << " " << vec[i].b << endl;

    return 0;
}
```

TASK 11-2 MAP

- ✓ A map is a container whose elements are pairs of a key and a value. When indexed by the key, a map returns the corresponding value.
- ✓ Note that the STL `<map>` would automatically sort the pairs of items by the keys.

```
// lab11-6.cpp
#include <iostream>
#include <map>
#include <string>
using std::cout;   using std::endl;
using std::map;    using std::string;

int main()
{
   map<int,string> classroom;

   classroom[9912345] = "Jacky";
   classroom[9923456] = "John";
   classroom[9934567] = "Mary";

   for (map<int,string>::const_iterator iter = classroom.begin();
      iter != classroom.end(); iter++)
   {
      cout << "ID: " << iter->first << " ";
      cout << "name: " << iter->second << endl;
   }

   return 0;
}
```

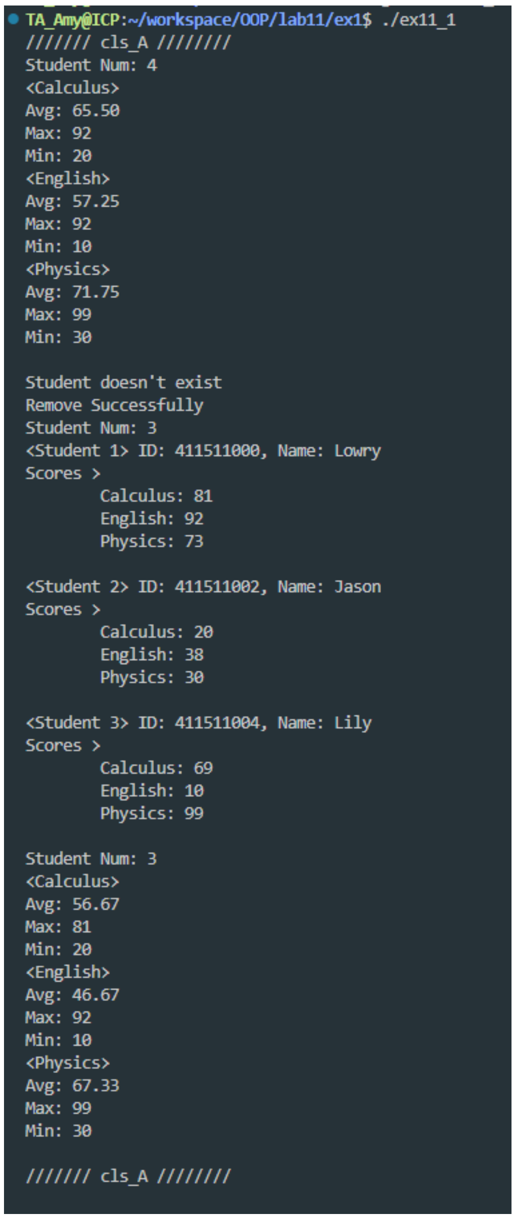
- ✓ Here is another example to use map:

```
// lab11-7.cpp
#include <iostream>
#include <map>
#include <string>
using std::cout;
using std::endl;
using std::map;
using std::string;

int main()
{
   map<string,int> age;
   map<string,int>::iterator it;
   age["Mary"] = 22;
   age["Jacky"] = 18;
   age["John"] = 20;
// practice_1
//   it = age.find("Jacky");
//   age.erase(it);

// practice_2
//   age.erase(age.find("John"));

// practice_3
//  cout << "Mary " << age.find("Mary")->second << endl;

    for (map<string,int>::const_iterator iter = age.begin();
        iter != age.end(); iter++)
    {
        cout << "name: " << iter->first << " ";
        cout << "age: " << iter->second << endl;
    }

    return 0;
}
```

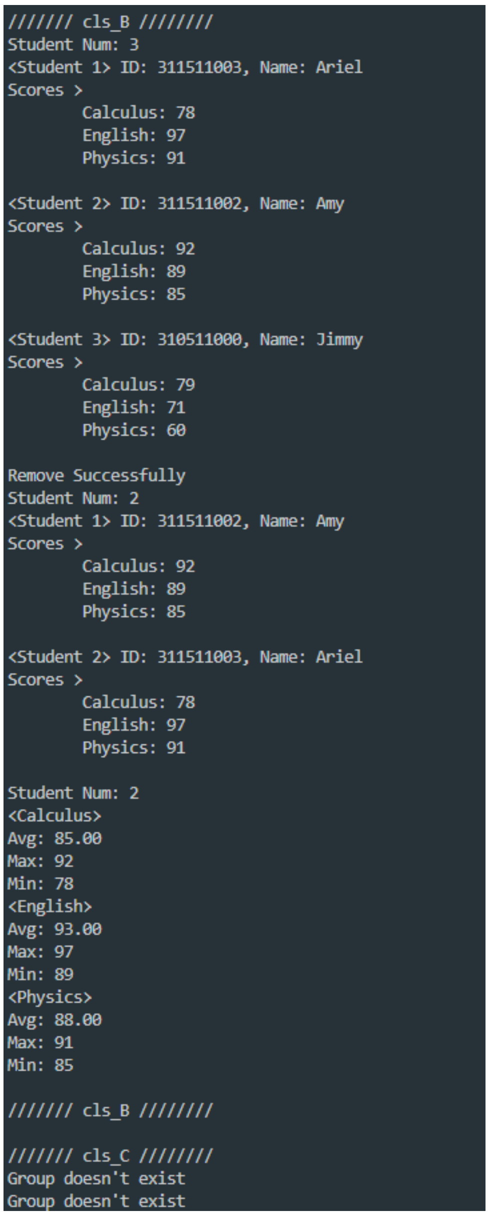
- ➢ Note that map stores elements in increasing order based on a less-than operation <

EXERCISE 11-1: EEE STUDY GROUP

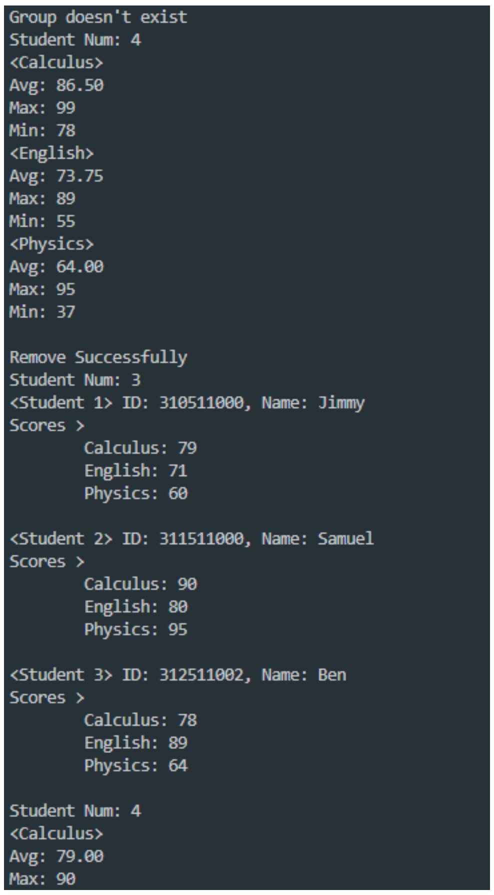
- ✓ Please finish the undefined function template in class `Student`, `Group`, and `EEE`.
- ✓ `main.cpp`, `Student.h`, `Group.h`, and `EEE.h` can get from: `/home/share/lab11/ex1`.
- ✓ Don't modify anything in `main.cpp`.
- ✓ The execution result:



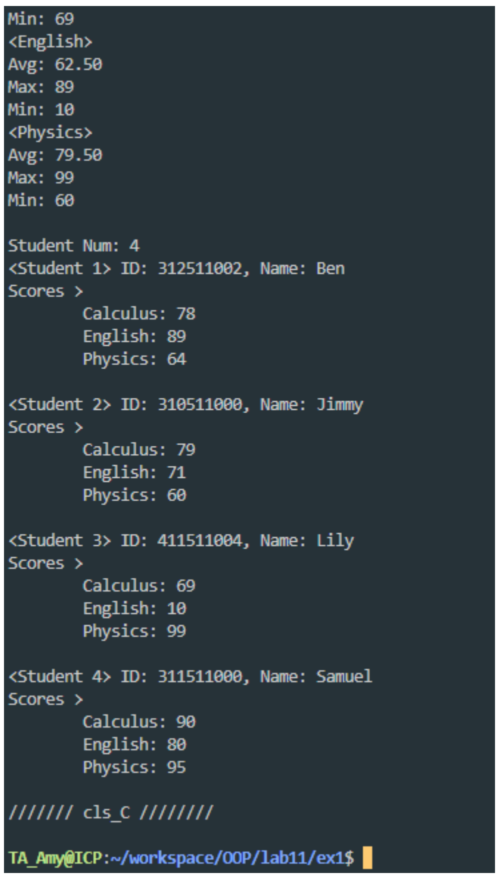
```
TA_Amy@ICP:~/workspace/OOP/lab11/ex1$ ./ex11_1
/////// cls_A ///////
Student Num: 4
<Calculus>
Avg: 65.50
Max: 92
Min: 20
<English>
Avg: 57.25
Max: 92
Min: 10
<Physics>
Avg: 71.75
Max: 99
Min: 30

Student doesn't exist
Remove Successfully
Student Num: 3
<Student 1> ID: 411511000, Name: Lowry
Scores >
        Calculus: 81
        English: 92
        Physics: 73

<Student 2> ID: 411511002, Name: Jason
Scores >
        Calculus: 20
        English: 38
        Physics: 30

<Student 3> ID: 411511004, Name: Lily
Scores >
        Calculus: 69
        English: 10
        Physics: 99

Student Num: 3
<Calculus>
Avg: 56.67
Max: 81
Min: 20
<English>
Avg: 46.67
Max: 92
Min: 10
<Physics>
Avg: 67.33
Max: 99
Min: 30

/////// cls_A ///////
```



```
/////// cls_B ///////
Student Num: 3
<Student 1> ID: 311511003, Name: Ariel
Scores >
        Calculus: 78
        English: 97
        Physics: 91

<Student 2> ID: 311511002, Name: Amy
Scores >
        Calculus: 92
        English: 89
        Physics: 85

<Student 3> ID: 310511000, Name: Jimmy
Scores >
        Calculus: 79
        English: 71
        Physics: 60

Remove Successfully
Student Num: 2
<Student 1> ID: 311511002, Name: Amy
Scores >
        Calculus: 92
        English: 89
        Physics: 85

<Student 2> ID: 311511003, Name: Ariel
Scores >
        Calculus: 78
        English: 97
        Physics: 91

Student Num: 2
<Calculus>
Avg: 85.00
Max: 92
Min: 78
<English>
Avg: 93.00
Max: 97
Min: 89
<Physics>
Avg: 88.00
Max: 91
Min: 85

/////// cls_B ///////

/////// cls_C ///////
Group doesn't exist
Group doesn't exist
```



```
Group doesn't exist
Student Num: 4
<Calculus>
Avg: 86.50
Max: 99
Min: 78
<English>
Avg: 73.75
Max: 89
Min: 55
<Physics>
Avg: 64.00
Max: 95
Min: 37

Remove Successfully
Student Num: 3
<Student 1> ID: 310511000, Name: Jimmy
Scores >
        Calculus: 79
        English: 71
        Physics: 60

<Student 2> ID: 311511000, Name: Samuel
Scores >
        Calculus: 90
        English: 80
        Physics: 95

<Student 3> ID: 312511002, Name: Ben
Scores >
        Calculus: 78
        English: 89
        Physics: 64

Student Num: 4
<Calculus>
Avg: 79.00
Max: 90
```



```
Min: 69
<English>
Avg: 62.50
Max: 89
Min: 10
<Physics>
Avg: 79.50
Max: 99
Min: 60

Student Num: 4
<Student 1> ID: 312511002, Name: Ben
Scores >
        Calculus: 78
        English: 89
        Physics: 64

<Student 2> ID: 310511000, Name: Jimmy
Scores >
        Calculus: 79
        English: 71
        Physics: 60

<Student 3> ID: 411511004, Name: Lily
Scores >
        Calculus: 69
        English: 10
        Physics: 99

<Student 4> ID: 311511000, Name: Samuel
Scores >
        Calculus: 90
        English: 80
        Physics: 95

/////// cls_C ////////

TA_Amy@ICP:~/workspace/OOP/lab11/ex1$
```

✓ main.cpp

```
#ifndef _GROUP_H_
#define _GROUP_H_
#include "Group.h"
#endif

#ifndef _STUDENT_H_
#define _STUDENT_H_
#include "Student.h"
#endif

#ifndef _EEE_H_
#define _EEE_H_
#include "EEE.h"
#endif

#include <iostream>
#include <vector>
using namespace std;

int main() {
   Student s1("311511000", "Samuel", vector<int>{90, 80, 95});
   Student s2("411511000", "Lowry", vector<int>{81, 92, 73});
   Student s3("310511000", "Jimmy", vector<int>{79, 71, 60});
   Student s4("311511001", "Brian", vector<int>{65, 82, 77});
   Student s5("311511002", "Amy", vector<int>{92, 89, 85});
```

```
    Student s6("310511001", "Sandy", vector<int>{60, 73, 94});
    Student s7("312511000", "Meg", vector<int>{49, 68, 77});
    Student s8("411511001", "Nicholas", vector<int>{88, 60, 92});
    Student s9("312511001", "Zack", vector<int>{99, 55, 37});
    Student s10("312511002", "Ben", vector<int>{78, 89, 64});
    Student s11("312511003", "Mark", vector<int>{32, 46, 88});
    Student s12("311511003", "Ariel", vector<int>{78, 97, 91});
    Student s13("411511002", "Jason", vector<int>{20, 38, 30});
    Student s14("411511003", "Larry", vector<int>{64, 99, 50});
    Student s15("410511000", "Bob", vector<int>{37, 88, 94});
    Student s16("410511001", "Nancy", vector<int>{50, 63, 83});
    Student s17("410511002", "Zoe", vector<int>{92, 95, 55});
    Student s18("410511003", "Rober", vector<int>{83, 74, 79});
    Student s19("410511004", "Frank", vector<int>{75, 31, 88});
    Student s20("411511004", "Lily", vector<int>{69, 10, 99});

    EEE A;
    cout << "/////// cls_A ////////" << endl;
    A.addStudent("cls_A", s5);
    A.addStudent("cls_A", s20);
    A.addStudent("cls_A", s2);
    A.addStudent("cls_A", s13);
    A.gradeDistribution("cls_A");
    A.removeStudentByID("cls_A", "311511001");  // s4
    A.removeStudentByID("cls_A", "311511002");  // s5
    A.viewInfo("cls_A", 1);
    A.gradeDistribution("cls_A");
    cout << "/////// cls_A ////////" << endl
        << endl;

    cout << "/////// cls_B ////////" << endl;
    A.addStudent("cls_B", s12);
    A.addStudent("cls_B", s5);
    A.addStudent("cls_B", s3);
    A.viewInfo("cls_B", 0);
    A.removeStudentByName("cls_B", "Jimmy");
    A.viewInfo("cls_B", 2);
    A.gradeDistribution("cls_B");
    cout << "/////// cls_B ////////" << endl
        << endl;

    cout << "/////// cls_C ////////" << endl;
    A.viewInfo("cls_C", 2);
    A.removeStudentByName("cls_C", "Zoe");
    A.gradeDistribution("cls_C");
    A.addStudent("cls_C", s10);
    A.addStudent("cls_C", s1);
    A.addStudent("cls_C", s3);
    A.addStudent("cls_C", s9);
    A.gradeDistribution("cls_C");
    A.removeStudentByName("cls_C", "Zack");
    A.viewInfo("cls_C", 1);
    A.addStudent("cls_C", s20);
    A.gradeDistribution("cls_C");
    A.viewInfo("cls_C", 2);
    cout << "/////// cls_C ////////" << endl
        << endl;

    return 0;
}
```

✓ `Student.h`

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

class Student {
   private:
    string id;
    string name;
    vector<int> Scores;  // store 3 score {Calculus, English, Physics}

   public:
    Student(string, string, vector<int>);  // (ID, Name, Scores)
    ~Student();
    friend ostream &operator<<(ostream &out, const Student a);

    // add any function you need
};
```

✓ `Group.h`

```
#include <string>
#include <vector>

using namespace std;
class Student;

class Group {
   private:
    vector<Student> students;
    int student_num;

    int sum_score_Calculus;
    int high_Calculus;
    int low_Calculus;

    int sum_score_English;
    int high_English;
    int low_English;

    int sum_score_Physics;
    int high_Physics;
    int low_Physics;

```

```
  public:
   Group();
   ~Group();
   void addStudent(Student);
   void removeStudentByID(string);
   void removeStudentByName(string);
   void gradeDistribution();
   // print out Avg, Max, Min Score of each subject

   // add any function you need
};
```

✓ `EEE.h`

```
#include <map>
#include <string>
#include <vector>

using namespace std;
class Student;
class Group;

class EEE {
  private:
   map<string, Group> groups;

  public:
   EEE();
   ~EEE();
   void addStudent(string, Student);          // add student to the group
   void removeStudentByID(string, string);    // remove student from the
                                                 group by ID
   void removeStudentByName(string, string); // remove student from the
                                                 group by Name
   void gradeDistribution(string);           // show grade distribution
                                                 of the group
   void viewInfo(string, int);          // view student info of the group,
                                          int represent diff mode
                                       // 0: sort by insertion time,
                                          1: sort by ID, 2: sort by name
};
```

✓ compile command:

```
g++ -o ex11_1 -o ex11_1 main.cpp Student.cpp Group.cpp EEE.cpp
```

EXERCISE 11-2: DICTIONARY

- ✓ In `ex11-2.cpp`, you should write a program to check whether the sentence can be formed by the words in the dictionary.
- ✓ For simplicity, every word is case-sensitive, that is, treat "you" and "You" as different words and count the number of each word independently.
- ✓ `D1.in` and `D2.in` can get from: `/home/share/lab11/ex2`.
- ✓ Hint: Use string as the key of the map and do the recursion.

- ✓ The output format is shown as following:
  - ➢ Test 1

```
$ ./ex11_2 D1.in catcatscatdooog
False

$ ./ex11_2 D1.in catcatscatdogsdog
cat: 2 (3->1)
cats: 1 (1->0)
dog: 1 (1->0)
dogs: 1 (1->0)
True
cat cats cat dogs dog

$ cat D1.in
cats 1
dog 1
cat 3
dogs 1
```

  - ➢ Test 2

```
$ ./ex11_2 D2.in IloveloveOOPandIlikeOPwenverymuch
I: 1 (1->0)
Il: 1 (2->1)
OP: 2 (2->0)
and: 1 (3->2)
ike: 1 (4->3)
love: 1 (3->2)
loveO: 1 (1->0)
much: 1 (2->1)
very: 1 (1->0)
wen: 1 (1->0)
True
I love loveO OP and Il ike OP wen very much

$ ./ex11_2 D2.in IloveOOPandIlikelikeOPwenverymuch
False
```

```
$ cat D2.in
I 1
Il 2
loveO 1
OP 2
and 3
ike 4
wenv 2
wen 1
very 1
much 2
love 3
```