Q1:

Map:

#include<iostream>

#include<string>

#include<map>

using namespace std;

int main() {

map<string, int> grades;

grades.insert(pair<string, int>("Felicity", 100));

grades.insert(pair<string, int>("Lily", 98));

grades.insert(pair<string, int>("Amy", 95));

grades.insert(pair<string, int>("Cindy", 93));

grades.insert(pair<string, int>("Lucy", 96));

for (map<string, int>::iterator itr=grades.begin(); itr!=grades.end(); itr++) {

cout << itr->first << " " << itr->second << endl;

}

return 0;

}

---

Amy 95

Cindy 93

Felicity 100

Lily 98

Lucy 96

---

Multimap:

#include <iostream>  
#include <string>  
#include <map>  
#include <list>  
  
using namespace std;  
  
int main()  
{  
 multimap<string,string> students;  
 students.insert(make\_pair("Felicity","100"));  
 students.insert(make\_pair("Lily","98"));  
  
 for (auto itr = students.begin(); itr != students.end(); ++itr)  
 {  
 cout << itr->first << '\t' << itr->second << '\n';  
 }  
  
}

---

Felicity 100

Lily 98

---

The map and multimap containers are specialized containers that should only be used when you are working with associative data. The map class is used when the keys in your application need to be unique and the multimap class is used when the keys in the application can be duplicates.

---

Q2：

1):

#include<iostream>

using namespace std;

int factorial(int n);

int main() {

int n;

cout << "Enter a positive integer: ";

cin >> n;

cout << "Factorial of " << n << " = " << factorial(n);

return 0;

}

int factorial(int n) {

if(n > 1)

return n \* factorial(n - 1);

else

return 1;

}

--

Enter a positive integer: 3

Factorial of 3 = 6%

---

2):

#include<iostream>

using namespace std;

int fib(int x) {

if ((x==1) || (x==0)) {

return(x);

} else {

return(fib(x-1) + fib(x-2));

}

}

int main() {

int x, i = 0;

cout << "Enter the number of terms of series: ";

cin >> x;

cout << "\nFibonnaci Sequence: ";

while(i < x) {

cout << " " << fib(i);

i++;

}

return 0;

}

---

Enter the number of terms of series: 5

Fibonnaci Sequence: 0 1 1 2 3%

---

Q3:

#include <iostream>

#include <set>

using namespace std;

struct classcomp {

bool operator() (const int& lhs, const int& rhs) const

{return lhs>rhs;}

};

int main() {

// Setup a standard set

set<int> a;

a.insert(5);

a.insert(1);

a.insert(6);

a.insert(3);

a.insert(7);

a.insert(2);

set<int>::iterator it;

for ( it=a.begin(); it != a.end(); it++ )

cout << " " << \*it;

cout << endl << endl;

return 0;

}

----

1 2 3 5 6 7

----

Q4:

Birthday.cpp

#include "birthday.h"  
#include <iostream>  
using namespace std;  
  
Birthday::Birthday(int y, int m, int d) : year(y), month(m), day(d) {}  
  
void Birthday::printDate() {  
 cout << year << "/" << month << "/" << day << endl;  
}

Birthday.h

#ifndef Q4\_BIRTHDAY\_H  
#define Q4\_BIRTHDAY\_H  
  
class Birthday {  
public:  
 Birthday(int y, int m, int d);  
 void printDate();  
  
private:  
 int year;  
 int month;  
 int day;  
};  
  
#endif //Q4\_BIRTHDAY\_H

Person.cpp

#include "birthday.h"  
#include "person.h"  
#include <iostream>  
#include <string>  
using namespace std;  
  
Person::Person(string n, Birthday b) : name(n), bd(b) {}  
  
void Person::printInfo() {  
 cout << name << endl;  
 bd.printDate();  
}

Person.h

#ifndef Q4\_PERSON\_H  
#define Q4\_PERSON\_H  
  
#include <string>  
using namespace std;  
  
class Person {  
public:  
 Person(string n, Birthday b);  
 void printInfo();  
  
private:  
 string name;  
 Birthday bd;  
};  
  
#endif //Q4\_PERSON\_H

Main.cpp

#include "birthday.h"  
#include "person.h"  
#include <string>  
  
int main() {  
 Birthday bd(1998, 1, 1);  
 Person p("Apple", bd);  
 p.printInfo();  
  
 return 0;  
}

Apple

1998/1/1