INM359 Object-Oriented Programming in C++

2023-24 Coursework

***To be done individually***

***Don’t “help” others – they can help themselves by studying the relevant material***

Contents

[Setting Up 1](#_Toc149828323)

[Standalone Questions 1](#_Toc149828324)

["Session 2 – Classes in C++ & Session 4 – Genericity, Containers" 1](#_Toc149828325)

["Session 3 – Overloading" 4](#_Toc149828326)

["Session 4 – Genericity, Containers" & "Session 5 – Pointers and Arrays Iterators" 5](#_Toc149828327)

[Part II – A simple phonebook app 6](#_Toc149828328)

[Attention to detail – Submitting 8](#_Toc149828329)

[Marking 8](#_Toc149828330)

# Setting Up

First, make sure you have a ***proper*** (command line-based) environment to develop and test your code. Use whichever IDE you want but your code ***MUST*** work with g++ version >=13.

Detailed instructions for installing a proper command-line environment are available at:

<https://www.staff.city.ac.uk/c.kloukinas/cpp/00-setup/cpp-command-line-environment-setup.html>

*You need to use and submit the Makefile from the above instructions!*

# Standalone Questions

## "Session 2 – Classes in C++ & Session 4 – Genericity, Containers"

1. Consider the following code. Fix the yellow parts, without modifying the parts in gray, so that the code compiles and runs correctly.

[3]

#include <iostream>

using namespace std; // File: q1.cc

class person {

int age;

public:

person(int a = 10) : age(a) {}

int get\_age() const;

void set\_age(int a);

};

/\* Your code here \*/

//

//

//

int main() {

person p1;

p1.set\_age(25);

cout << p1.get\_age() << endl; // should print 25

return 0;

}

1. Consider this slightly modified version of the person class – fix it by adding code in the yellow parts.

[2]

#include <iostream>

using namespace std; // File: q2.cc

class A {

public:

A(int i) {};

};

class person {

int age; A an\_a;

public:

//

person(int a) : age(a), an\_a(a) {}

int get\_age() const;

void set\_age(int a);

};

//

//

//

int main() {

person p1;

p1.set\_age(25);

cout << p1.get\_age() << endl; // should print 25

return 0;

}

1. One more person class variant – this time a generic class. Fix the yellow parts, without modifying the parts in gray, so that the code compiles and runs correctly.

[3]

#include <iostream>

using namespace std; // File: q3.cc

template <typename T>

class person {

T age;

public:

//

person(T a) : age(a) {}

T get\_age() const;

void set\_age(T a);

};

/\* Your code here \*/

//

//

//

int main() {

person<unsigned char> p1;

p1.set\_age(25);

cout << (int) p1.get\_age() << endl; // should print 25

return 0;

}

1. Now consider this final person class variant. Complete the main so that it prints 25 after having printed 10, without adding more methods to class person or changing the visibility of its members.

[3]

#include <iostream>

using namespace std; // File: q4.cc

class person {

int age;

public:

person(int a = 10) : age(a) {}

int get\_age() const;

};

/\* Your code here \*/

int main() {

person p1;

cout << p1.get\_age() << endl; // should print 10

/\* Your code here \*/

cout << p1.get\_age() << endl; // should print 25

return 0;

}

1. Consider the following use of class my\_array\_class. Fix the code in the class so that the grayed-out code compiles and runs correctly.

**Note:** you should not modify any of the code in gray.

[1]

#include <iostream>

using namespace std; // File: q5.cc

class my\_array\_class {

size\_t len = 3;

int \*a = new int [3];

public:

my\_array\_class() { a[0] = 1; a[1] = 2; a[2] = 3; }

my\_array\_class(size\_t ln, const int \*o) : len(ln), a(new int [ln])

{ for (size\_t n=0; n<ln; ++n) a[n] = o[n]; }

~my\_array\_class() { delete[] a; }

size\_t get\_length() { return len; }

int get(size\_t n) {return a[n];}

int set(size\_t n, int v) { int tmp = a[n]; a[n] = v; return tmp; }

};

void foo( const my\_array\_class & a2, size\_t i ) {

if (i < a2.get\_length())

std::cout << a2.get(i) << std::endl;

}

int main() {

int zero12[] = {13, 1, 2};

my\_array\_class a1(3, zero12);

foo(a1, 0); // should print 13

return 0;

}

## "Session 3 – Overloading"

1. Extend the code in the class my\_array\_over below so that the grayed-out code compiles and runs correctly. (**Hint:** compare the grayed-out code here with Q5’s.)

[2]

#include <iostream>

using namespace std; // File: q6.cc

class my\_array\_over {

size\_t len = 1;

int \*a = new int [1];

public:

my\_array\_over() { a[0] = 0; }

my\_array\_over(size\_t ln, const int \*o) : len(ln), a(new int [ln])

{ for (size\_t n=0; n<ln; ++n) a[n] = o[n]; }

~my\_array\_over() { delete[] a; }

/\* Put your code here \*/

};

void foo( const my\_array\_over & a2, size\_t i ) {

if (i < a2.get\_length())

std::cout << a2[i] << std::endl;

}

int main() {

int zero12[] = {23, 1, 2};

my\_array\_over a1(3, zero12);

a1.set(0, 13);

foo(a1,0); // should print 13

return 0;

}

1. Now do the same as in Q6 for the following code.

[2]

#include <iostream>

using namespace std; // File: q7.cc

class my\_array\_over2 {

size\_t len = 1;

int \*a = new int [1];

public:

my\_array\_over2() { a[0] = 0; }

my\_array\_over2(size\_t ln, const int \*o) : len(ln), a(new int [ln])

{ for (size\_t n=0; n<ln; ++n) a[n] = o[n]; }

~my\_array\_over2() { delete[] a; }

/\* Put your code here \*/

};

void foo( const my\_array\_over2 & a2, size\_t i ) {

if (i < a2.get\_length())

std::cout << a2[i] << std::endl;

}

int main() {

int zero12[] = {13, 1, 2};

my\_array\_over2 a1(3, zero12);

**a1[1] = 14; /\* EXTRA LINE!!! \*/**

foo(a1,1); // should print 14

return 0;

}

1. Consider the class m2dvector below, used for holding 2-dimensional (mathematical) vectors.
   1. Overload the output operator, so that you can print objects of class m2dvector.

[6]

* 1. Overload the input operator, so that you can read objects of class m2dvector.

[12]

#include <iostream>

#include <string>

#include <sstream>

#include <vector>

#include <exception>

using namespace std; // File: q8.cc

class m2dvector {

vector<int> vi = {0, 0};

public:

m2dvector(vector<int> some\_vi) : vi(some\_vi) {if (some\_vi.size() > 2) throw std::invalid\_argument("Input vector should have size 2."); }

const vector<int> & get\_data() const { return vi; }

};

/\* Put your code here \*/

int main() {  
 try {

vector<int> vi = {11, 12, 13, 14, 15};

m2dvector z1({vi[0], vi[1]});

m2dvector z2({vi[3], vi[4]});

cout << z1 << endl; /\* should print: <11 12> \*/

cout << z2 << endl; /\* should print: <14 15> \*/

string s1 = "<11 12> <13 14>a"; /\* s1 is a \*single\* test – code   
 must work in general \*/

istringstream iss(s1);

iss >> z2 >> z1;

cout << z1 << endl; /\* should print: <13 14> \*/

cout << z2 << endl; /\* should print: <11 12> \*/

char c;

iss >> c; cout << c << endl; /\* should print ‘a’ – the char after the 2nd m2dvector above. \*/

ostringstream oss;

oss << z2;

string so = oss.str();

for ( char c : so )

if (c == '\n') {cout << "Output contains a newline\n"; return 0;}

cout << "Output does not contain a newline\n";  
 } catch (...) { throw; }

return 0;

}

## "Session 4 – Genericity, Containers" & "Session 5 – Pointers and Arrays Iterators"

1. Consider the following code – complete it (in the yellow area) so that it compiles and runs.

[12]

#include <iostream>

#include <map>

#include <string>

#include <algorithm>

using namespace std; // File: q9.cc

template <typename X, typename Y>

class class1 {

map<X, Y> map1;

public:

class1(map <X, Y> some\_map);

const map <X, Y> & get\_data() const;

};

/\* a) Put your code here to get it to compile \*/

int main() {

map<int, float> mi2f = { {1, 11}, {2, 12}, {3, 13}};

class1<int, float> c1(mi2f);

const map<int, float> & c1dt = c1.get\_data();

/\* b) Put your code here to print the elements of c1dt. \*/

return 0;

}

# Part II – A simple phonebook app

The following questions consider the case where we need to develop a simple phone book.

1. Develop a class person (in **qpersons.cc**) that offers get\_name() methods (no need to split into forename/surname – a single name will do).

[5]

1. Develop two classes person\_with\_phone and person\_with\_email (in **qpersons.cc**), that each inherits from person, with appropriate get\_phone() and get\_email() methods.

[5]

1. Develop a class person\_with\_phone\_email (in **qpersons.cc**) that inherits from both person\_with\_phone and from person\_with\_email.

[10]

1. Appropriately overload operator<< (in **qpersons.cc**) so that you can print ***any*** type of person. The format to be used is:

"<person N " name[" T " telephone]?[" E " email]? " >"

The notation "[a]?" means that "a" may be present or absent. The output ends with a space (" ") followed by the greater than (">") symbol.

Examples ***(use them all)***:

<person N TomJones >

<person N DickTracey T +49.921.1434 >

<person N HarryPotter E hpotter@gmail.com >

<person N MaryJones T +39.921.1434 E mjones@gmail.com >

<person N JohnSmith T +33.921.1434 E jsmith@gmail.com >

***Another example you should use is the output of the following commands that you should run on a Bash shell (copy-paste them, don't type them):***

function doit() {

sleep 1; clear

echo 'Type your City login (e.g., abcd123) in lowercase followed by return:' >&2

read lg

lg=`echo ${lg} | tr '[A-Z]' '[a-z]'`

lgmd5=`echo ${lg} |md5sum |sed -e 's/[ -]\*$//'`

lgmd5c=`echo ${lgmd5} | tr -d '[0-9]'`

lgmd5n=`echo ${lgmd5} | tr -d '[A-Za-z]'`

echo '<person N '$lgmd5c T 1$lgmd5n' >'  
}

doit

# Note the "doit" above!

*Note that in theory a base class (like person) shouldn't be aware of the existence of its derived classes (like person\_with\_X) – try to consider this when designing your solution.*

[15]

* 1. Use all the examples above (the one from doit as well) to test the operator<< inside your main.
  2. Make the destructors of all person classes print the object to cerr.

1. Implement a class contacts (in **qpersons.cc**) that holds all your contacts (that are some kind of person). You can assume that two different (in real life) persons will always have a different name. So:
   * Redefine the operator== for all kinds of persons, so that it checks whether their names (alone) match.
   * Also overload the operator< so that it checks whose name comes first alphabetically.
   * Add the persons (method add) in the examples in Q13 into your contacts (the one obtained from running doit too). First add them with only their names, and then add them again with their extra details (see the note that follows).
   * Print the contents of your contacts (method print\_all).
   * Use the standard algorithm find\_if to implement a method print\_all\_after, which takes a string argument and prints all contacts whose name is greater than that argument.

**NOTE:** You add a person in your contacts using the member function add(). This should work even if that person is already in the contacts (useful when you want to update their contact details). So, the following should *update* John’s details – note how the person is being passed to add:

contacts c;

person john1("JohnSmith");

c.add(& john1);

person\_with\_phone john2("JohnSmith", "+33.02.0101.0202");

c.add(& john2);

This example shows why we can't store person objects into the container directly – we can't replace a person object with a person\_with\_X one (try it to see what you get).

[16]

1. Change your code in Q14, so that it uses smart pointers (shared\_ptr<T>).

[6]

1. Create a folder called q16 and split your code from Q15 into separate header (person.h, person\_with\_....h, contacts.h) and source files (person.cc, person\_with\_....cc, contacts.cc) inside q16/ and get it to compile. Header files should contain just the method declarations, while source files should contain the method definitions. Function main should be in the file q16/qmain.cc

[7]

*(Can you extend the Makefile so that the minimum number of files are recompiled whenever one of them is modified? Extra Karma points!)*

1. The destructors of all persons print the object to cerr (see Q13.2) – explain why the output is what it is (in file explanation.txt – include the ***entire*** output, as produced by the program). Identify the lines where each object was created and match them with the destructor outputs (in the vi editor, “:set nu” enables numbered lines so you can copy-paste the source of the main with lines).

[10]

# Attention to detail – Submitting

You need to submit \***code**\* *(source code, not executables!)*, *not* a report in Word/PDF/etc.

So, create a folder named after your Moodle login (abcd123 – in lowercase) and **copy** in there the files q1.cc, q2.cc, …, q9.cc, qpersons.cc, qmain.cc and any other files from Q16, along with the Makefile from page 1 (described in the environment setup instructions web page).

Zip your folder *(use zip, not rar/tar/7zip/etc.!)* – on the terminal that's done with:

zip -r abcd123.zip abcd123/

Unzip your zip file **somewhere else** (e.g., /tmp):

cp abcd123.zip /tmp ; cd /tmp ; unzip abcd123.zip ; cd abcd123

Try to compile all your files there – do they compile with the provided Makefile (hopefully with no errors/warnings)?

If so, submit your zip file (which should contain the Makefile and the source files only – no binary code!).

**5%**

# Marking

There are 120 marks overall. These amount to 95% of the coursework mark (5% is for attention to detail).

Of these 120 marks, the first 45 count as a percentage, while the rest are weighted – see below.

cat > marks2023.cc << EOF

#include <iostream> // std::cout

#include <algorithm> // std::min

using namespace std;

int main() {

float marks;

const float threshold = 45.0;

const float mx = 120.0;

const float weight = (95.0 - threshold) / (mx - threshold);

while (cin >> marks)

cout // using the ite (?:) operator (ite = if-then-else)

<< ((marks<=threshold) ? marks : (threshold+(marks-threshold)\*weight))

<< char(9) // or, using min & max:

<< max( min(marks, threshold), min(marks, threshold) + (marks-threshold)\*weight )

<< endl;

return 0;

}

EOF

make marks2023

./marks2023