COMP3400-R-30
2019 Winter, Feb. 5, 2019 in TC 204
University of Windsor, School of Computer Science

Midterm 1 Examination Solutions

Instructor: Mr. Paul Preney

Student ID:	
FIRST Name:	
LAST Name:	

"I have neither given nor received unauthorized help with this examination.
Any suspicion of cheating will automatically void my mark on this examination."

Signature
Unsigned examination booklets will not be graded.
Signature implies agreement with the above statement in quotes.

INSTRUCTIONS:

1. You have **1.25 hours** maximum to complete this examination. Pace yourself accordingly.
2. Write your answers in the space provided. No additional space will be provided.
3. Do **not** remove any papers from this booklet or add new ones.
4. You may **not** use any reference materials or books.
5. Ensure that you have all **13 pages** of this examination (including this page) before starting to write this exam. If you don't bring this to the attention of the instructor immediately.
6. Write/print legibly: illegible answers (or portions thereof) will not receive marks.

SCORE: ____________ / 58

Part I

Multiple Choice and Short Answer Questions

For each question in this section, neatly and clearly **circle, check, or underline** the **single best response** which most correctly completes/answers the statement/question given for multiple choice or true/false style questions, otherwise write in the appropriate answer(s) in the space provided appropriate for that question. Read carefully! Unintelligible or ambiguous responses will receive a mark of zero (0) for that question, so ensure that your answer is clear.

1. (1 point) What is today called C++ was created at Bell Labs in 1979 by ______________. (Provide the person's first and last name in your answer.)

   1. **Bjarne Stroustrup**

2. (1 point) The ISO document number of the various C++ standards is ______________:
   - ◯ ISO/IEC 9899
   - ◯ ISO/IEC 9989
   - [✓ **ISO/IEC 14882**]
   - ◯ ISO/IEC 15992
   - ◯ ISO/IEC 19989

3. (1 point) C++ is a language and a complete system.
   - ◯ True
   - [✓ **False**]

4. (1 point) The design of C++ is to prevent all misuses of a useful feature.
   - ◯ True
   - [✓ **False**]

5. (1 point) The design of C++ is to permit no implicit violations of the static type system.
   - [✓ **True**]
   - ◯ False

6. (1 point) The zero-overhead rule means, "What you don't use, you don't pay for."
   - [✓ **True**]
   - ◯ False

7. (1 point) The imperative programming paradigm defines computation in terms of programming statements that describe changes in ______________.

   7. **state**

8. (1 point) What the modular, object-based, and object-oriented programming paradigm all have in common is that they allow for ______________, i.e., the ability to hide functions/types defined within.

   8. **encapsulation**

9. (2 points) Briefly describe how the object-based programming paradigm and the the object-oriented programming paradigm are distinguished from one another.

   Solution: The object-based paradigm does not support inheritance whereas the object-oriented paradigm does.

10. (3 points) **Explain** what one has to do to include C headers such as `<ctype.h>` in C++. Be sure to also write the C++ include directive code for `<ctype.h>` in your answer.

    Solution: For a given C header, one must remove the `.h` extension and prefix `c` in front of the header name to include it in C++.

    include-cctype.cxx

    ```
    #include <cctype>
    ```

11. (2 points) If a C++ compiler chose to implement a variable of type `int*&` as a pointer, what would the variable's *exact* equivalent type be (expressed as a C++ pointer type)?

    Solution: `int * * const`

12. (2 points) If a C++ compiler chose to implement a variable of type `double const&&` as a pointer, what would the variable's *exact* equivalent type be (expressed as a C++ pointer type)?

    Solution: `double const * const`

13. (1 point) Given the declaration, `unsigned u;`, what does `--u;` return?

    ◯ an rvalue

[✓ **an lvalue**]

◯ neither of these

14. (1 point) Given the declaration, `signed s;`, what does `s--` return?

[✓ **an rvalue**]

◯ an lvalue

◯ neither of these

15. (1 point) Excluding results of casts, in practice a variable can be determined to be an **rvalue** when it ______________.

15. **doesn't have a name**

16. (1 point) Excluding results of casts, in practice a variable can be determined to be an **lvalue** when it ______________.

16. **has a name**

17. (3 points) A coworker has written the line of C++11 code "`auto v;`" which clearly won't compile. The coworker has asked you (i) why this line won't compile and (ii) how might it be fixed. Write your answers to (i) and (ii) to the coworker below.

Solution:

i The line won't compile since a variable declared with `auto` **must** be assigned to an expression at the time of declaration (so that the expression's resulting type becomes the type of the variable). (Without a type, the C++ compiler cannot declare the variable.)

ii The fix requires the coworker to either explicitly write in the type of the variable instead of using `auto`, or, set the value of the variable at the time of declare to an expression (when using `auto`). e.g., `int v = 23;` e.g., `auto v = 23;`

18. (2 points) When using IOStreams objects, e.g., `std::cin`, Prof. Preney often used the stream object in contexts requiring a `bool` value, e.g., `while (cin >> i) /* code */;` and `if (cin) /* code */;`. Clearly an IOStream object is **not** a `bool` and yet the code compiles. Explain what the results of such boolean contexts are (i.e., what triggers a `true` value and what triggers a `false` value) when an IOStream object is used in this way.

Solution: When an IOStream object is cast to a `bool` it returns the result of `object.good()`, i.e., `true` if no errors, failures, or end-of-file has yet occurred on the stream; or `false` otherwise.

19. (2 points) Describe the memory layout of class data members when all of the classes involved only use single inheritance to (singly) inherit from one another.

Solution: The data members are concatenated together (with padding and/or alignment as is appropriate).

ASIDE 1: The ISO standard guarantees that within each class all data members with the same scope access level are placed in memory in order of appearance, each at a higher memory address. The relative ordered of data members having different scope access is unspecified.

ASIDE 2: Technically the ISO standard states that the order in which base class subobjects are allocated in the most derived object is unspecified. In addition, a base class subobject might have a different layout from the layout in a most derived object of the same type (e.g., due to empty base optimization).

20. (2 points) You are tasked with translating code from another object-oriented programming language, e.g., Java, to C++. In C++ all **abstract** classes must be implemented as `class`es that ______________. (Your answer must finish this statement appropriately in a way that guarantees the C++ `class` is an abstract `class`.)

Solution: "each contain at least one **pure virtual** function."

21. (2 points) In C++ when virtual appears in front of a member function, what is the minimal overhead involved in calling the member function when it is invoked and why?

Solution: The member function will be called indirectly via a function pointer. (Additional indirections may be required depending on how such is implemented.) This is necessary to implement run-time polymorphism of OOP methods.

22. (2 points) In C++ when virtual appears in front of an inherited class, what is the minimal overhead involved in accessing any data member within that class (or any of its parents) and why?

Solution: Accessing a data member of that class or any of its parents requires at least one indirection to access the **single** instance of that class (or any of its base classes) at that point in the hierarchy.

Part II

General Questions

Answer all parts of each question in the space provided below each question. You are expected to answer questions using complete sentences and proper grammar. If the answer has program code/diagrams, write the code fragment(s) or the diagram portion(s) that answers the question **unless you are explicitly asked to write a full-and-complete program or diagram.** Your answers can assume `using namespace std;` has been asserted somewhere above your code.

1. (2 points) Write the code fragment to declare a variable called **v** representing a `std::vector` container of `double` elements.

Solution: `vector<double> v;`

2. (3 points) Write the code fragment to read in all `double` values from `std::cin` using a single `while` loop into the variable **v** you defined in the previous question. (Any stream error, failure, or end-of-file must terminate the `while` loop.) (Remember to declare your `double` variable!)

Solution:

while-read-double.cxx

```
double d;
while (cin >> d)
  v.push_back(d);
```

3. (6 points) Using a traditional C++98-style **for** loop, write the code fragment to output all **double**s stored in **v** read in the previous question to std::cout. Ensure there is a space between each **double** and a newline character output after the for loop finishes. (Hint: You will want to use **auto** to declare any needed variables, begin(), and end() in your answer.)

Solution:

traditional-for-v.cxx

```
// strictly speaking the question required a space between doubles
// so this is a solution for such...
for (auto i=begin(v), iEnd=end(v); i != iEnd; ++i)
{
  cout << *i;
  if (next(i) != iEnd)  // or: if (i != iEnd-1)  ... since v is a vector
    cout << ' ';
}
cout << '\n';
// but this slightly relaxed version was accepted...
for (auto i=begin(v), iEnd=end(v); i != iEnd; ++i)
  cout << *i << ' ';
cout << '\n';
// as well as this one (since v is a vector)...
for (std::vector<double>::size_type i=0; i != v.size(); ++i)
  cout << v[i] << ' ';
cout << '\n';
```

4. (3 points) The previous question used a traditional C++98-style **for** loop. Write the answer to the previous question using the range-based **for** loop introduced in C++11 instead of the traditional **for** loop.

Solution:

range-for-v.cxx

```
// strictly speaking the question required a space between doubles
// so this is a solution for such...
auto len = v.size();
for (auto e : v)
  cout << e;
  if (--v != 0)
    cout << ' ';
}
cout << '\n';
// but this slightly relaxed version was also accepted...
for (auto e : v)
  cout << e << ' ';
cout << '\n';
```

5. (10 points) The previous questions read in a `std::vector` of **doubles** stored in a variable called **v**. The task in this question is to call three algorithms:

   - `std::sort()` to sort the entire vector,
   - `std::lower_bound()` to find the first positive value (i.e., the first value greater than or equal to 0.0), and then call,
     - Hint: Use **auto** to declare the variable that stores the result of the call. You will need to use this variable in the next call below.
   - `std::count_if()` to determine the number of values **less than 10.0 or greater than 50.0** starting at the first positive value found previously with `std::lower_bound()`.
     - Hint: Use a lambda function as your predicate.
     - Hint: Since you don't know the integer type `std::count_if()` returns, use **auto** to declare the variable that will store the integer count returned.

   Finally output the number of values less than 10.0 or greater than 50.0 returned by the `std::count_if()` call to `std::cout`.

   Here are the prototypes and brief descriptions of these calls:

   algorithm-info.cxx

```
template <typename RndIter>            // RndIter must be random-access iterator
void sort(RndIter first, RndIter last); // sorts [first,last) range

template <typename FwdIter, typename T> // FwdIter must be a forward iterator
FwdIter lower_bound(FwdIter first, FwdIter last, T const& value);
  // lower_bound returns the iterator pointing to the first element
  // in the range [first,last) that is NOT LESS THAN value, or,
  // last if no such element is found.

template <typename InputIter, typename UnaryPredicate>
INTEGER_TYPE count_if(InputIter first, InputIter last, UnaryPredicate func);
  // NOTE: INTEGER_TYPE is some integer type (Hint: Use auto to deal with this.)
  // count_if returns the number of elements in the range [first,last)
  // where it counts the number of elements for which func returns true.
```

   NOTE: Do not optimize the code so it is more efficient. Simply perform the calls in the order listed as instructed.

   Do not write a complete program, `main()`, or include directives. Simply write the four (4) C++ statements to perform the above operations, i.e., one statement for each of the three algorithm calls (in that order) and one for the cout statement.

   Solution:

   algorithm-tasks.cxx

```
// NOTE: This is a complete program for exposition purposes only.
// SAMPLE RUN:
```

```
//    2 -3 -4 4 3 4 65 76 2 43 6 30 20 40
//    There are 8 values <= 10.0 and >= 50.0.

#include <vector>
#include <algorithm>
#include <iostream>

using namespace std;

int main()
{
  vector<double> v;

  double d;
  while (cin >> d)
    v.push_back(d);

  // START OF REQUIRED ANSWER
  sort(begin(v), end(v));

  auto pos = lower_bound(begin(v), end(v), 0.0);

  auto quantity = count_if(
    pos, end(v),
    [](auto const& val)
    {
      return val <= 10.0 || val >= 50.0;
    }
  );

  cout << quantity << '\n';
  // END OF REQUIRED ANSWER
}
```

This page has been left intentionally blank.

This page has been left intentionally blank.

This page has been left intentionally blank.