Sample Exam Week 09

CSE 232 (Introduction to Programming II)

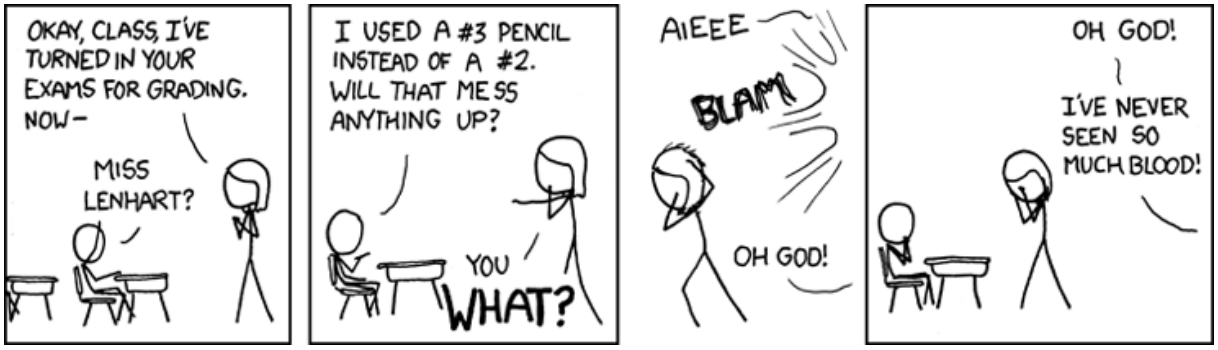
VERSION A

Full Name: .....

Student Number: .....

Instructions:

- DO NOT START/OPEN THE EXAM UNTIL TOLD TO DO SO.
- You may however write and bubble in your name, student number and exam **VERSION/FORM NUMBER** (with a #2 pencil) on the front of the printed exam and bubble sheet prior to the exam start. This exam is Version A. Your section doesn't matter and can be ignored.
- Present your MSU ID (or other photo ID) when returning your bubble sheet and printed exam.
- Only choose one option for each question. Please mark the chosen option in both this printed exam and the bubble sheet.
- Assume any needed `#include`s and `using std::...;` namespace declarations are performed for the code samples.
- Every question is worth the same amount of points. There are 55 questions, but you only need 50 questions correct for a perfect score.
- No electronics are allowed to be used or worn during the exam. This means smart-watches, phones and headphones need to be placed away in your bag.
- The exam is open note, meaning that any paper material (notes, slides, prior exams, assignments, books, etc.) are all allowed. Please place all such material on your desk prior to the start of the exam, (so you won't need to rummage in your bag during the exam).
- If you have any questions during the exam or finish the exam early, please raise your hand and a proctor will attend you.



http://xkcd.com/499/

1. The `std` in `std::cout` is an example of what part of C++?

   (a) A namespace
   (b) An exception
   (c) A variable
   (d) A file
   (e) A class
   (f) A function
   (g) A using
   (h) An object

2. The definition of a publicly accessible function template should be put in what file?

   (a) The header file
   (b) The main file
   (c) The template file
   (d) The implementation file

3. What is output from this code?

```
template<typename T>
void func(T const & arg) {
  cout << arg << '!';
}
int main() {
  func(3.5);
  func(3 == 5);
  func(string("hi"));
}
```

   (a) The code generates a run-time error.
   (b) The code generates a syntax error.
   (c) `3.5!false!hi!`
   (d) `3.5!0!hi!`
   (e) `3.5!\nfalse!\nhi!\n`
   (f) None of the above.

4. What is the difference between lambda expressions and named functions?

   (a) Lambdas must be defined in implementation files.
   (b) Lambdas can have capture lists.
   (c) Lambdas are used as arguments to algorithms, functions are not.
   (d) Lambdas always copy their arguments, functions can use references and pointers to avoid copying.
   (e) Lambdas always take two parameters, functions can take 0 or more parameters.
   (f) There is no difference between a lambda and a named function.

5. Which of the following have access to all private member attributes?

   (a) Const member functions of other classes
   (b) `operator>>`
   (c) Generic algorithms
   (d) The `main()` function
   (e) None of the above

6. Making a class templated allows which of the following capabilities to be possible?

   (a) A matrix class that can have many different numbers of rows and columns
   (b) A container class that can hold different types of elements
   (c) A class that can be written to different output streams
   (d) A string class that can hold many different letters, digits, and whitespace characters
   (e) All of the above

7. In the lab where you wrote the `fill_random` member function of `Table`, why was it described as a bad idea to use `std::random_device` repeatedly to generate random numbers?

   (a) Because it is slow.
   (b) Because it depletes sources of randomness available to other processes.
   (c) Both of the above.
   (d) None of the above (this is the recommended technique).

8. Which of the following is true about templates?

   (a) Parameterized types are required to call function templates
   (b) Templates allow a class to hold many elements
   (c) Function objects, like Less-than from Section 7.3.2, are function templates.
   (d) Only function templates can access private attributes of a class
   (e) None of the above are true

9. Is the following legal?

```
template <typename T>
class Node {
 public:
  T data_;
 private:
  Node<T> * next_;
};
```

   (a) No, a class must have function members.
   (b) No, a data member cannot be a pointer.
   (c) No, it has a syntax error.
   (d) No, a type cannot refer to itself.
   (e) Yes, it is legal.

10. What special privileges does a friend function have over a class?

    (a) It can call deleted functions.
    (b) It can make public members private.
    (c) It can define the `operator<<`.
    (d) It can alter const objects.
    (e) None of the above.

11. Where should library function templates be specified?

    (a) Templated functions must be put in a template file (extension .tmp)
    (b) In a header file
    (c) In both a header and implementation file
    (d) In a implementation file

12. Why should you often make data members private?

    (a) To enforce encapsulation via access control.
    (b) Because data members have to be private in a class.
    (c) Because friend functions can only access private data members.
    (d) Because getters and setters can only work on private data members.
    (e) None of the above.

13. What is the primary benefit for using lambda expressions?

    (a) To be able to define a single-use function right where it is needed.
    (b) Because lambda functions can access private data members.
    (c) Because lambda functions can have multiple parameters.
    (d) Because function templates require lambda expressions to work.
    (e) None of the above.

14. What is the primary benefit to writing function templates?

    (a) So that the function runs faster.
    (b) So that the function compiles faster.
    (c) So that the function is throws exceptions.
    (d) So that the function is generic.
    (e) So that the function is composited.
    (f) None of the above.

15. Which of the following expressions have type `int`, assuming `vec` is a 2-dimensional table with 3 rows and 2 columns and has type `vector<vector<int>>`.

    (a) `vec.size()`
    (b) `vec[1].size()`
    (c) `vec[1][0]`
    (d) `vec[0]`
    (e) Two of (a-d) have type `int`.
    (f) Three of (a-d) have type `int`.
    (g) Four of (a-d) have type `int`.

This page intentionally left blank.