Sample Exam Week 09

CSE 232 (Introduction to Programming II)

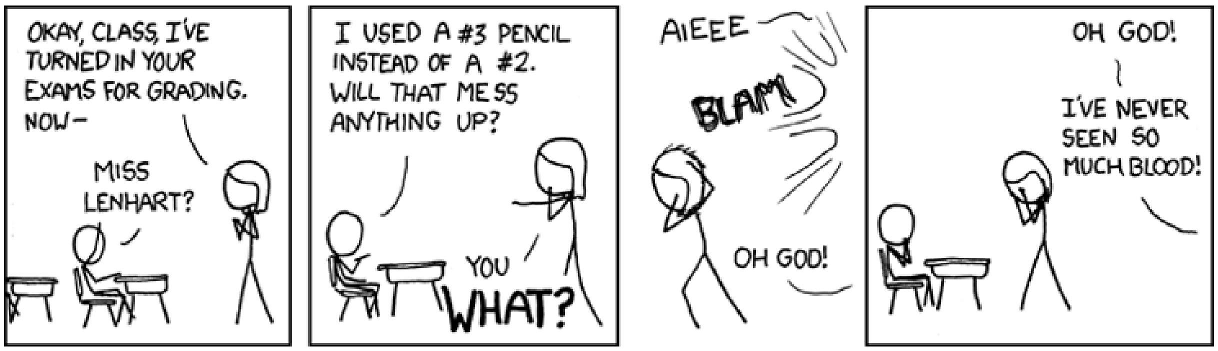
VERSION A

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.


OKAY, CLASS, I'VE TURNED IN YOUR EXAMS FOR GRADING. NOW–
MISS LENHART?
I USED A #3 PENCIL INSTEAD OF A #2. WILL THAT MESS ANYTHING UP?
YOU WHAT?
AIEEE
BLAM
OH GOD!
OH GOD!
I'VE NEVER SEEN SO MUCH BLOOD!

http://xkcd.com/499/

1. What will the following code output?

```
std::vector<int> v{1,2,3,4};
for (auto it = v.begin() + 1;
        it != v.end();
        it++) {
    std::cout << *it;
}
```

   (a) `1234`
   (b) `234`
   (c) `123`
   (d) `23`
   (e) `1`
   (f) `2`

2. When are the begin and end iterators of a vector pointing at the same position?

   (a) When the vector has just been resized.
   (b) When the vector has exactly one element
   (c) When the vector has been passed to an algorithm
   (d) When the vector is empty
   (e) Never, the end iterator always points after the begin iterator

3. When using a range-based for loop on a `std::map<int, std::string>`, what type is each element returned as?

   (a) `string`
   (b) `auto`
   (c) `std::pair<int, std::string>`
   (d) `int`
   (e) `std::map<int, std::string>`

4. Which of the following does **NOT** return the last element of a vector (of size 10) named x?

   (a) `x.back()`
   (b) `x[9]`
   (c) `x.at(9)`
   (d) `x[x.size() - 1]`
   (e) `x.end()`
   (f) Multiple of the above will not work.
   (g) All of the above will work.

5. Which of the following types can be iterated over with a range-based for loop (i.e. a for-each loop)?

   (a) `string`
   (b) `int`
   (c) `double`
   (d) `map<int, string>`
   (e) `vector<int>`
   (f) All of the above
   (g) (a), (d), and (e)
   (h) (d) and (e)
   (i) (a) and (e)

6. What is the difference between
`for (auto x : xs) ...`
and
`for (auto const & x : xs) ...`

   (a) The first range-based for loop copies each element.
   (b) Only the second range-based for loop can work if `xs` is const.
   (c) Only the first loop will compile if `x` in a fundamental type.
   (d) They have the same behavior if if `xs` is a vector of ints, but not a vector of strings.
   (e) The second range-based for loop is able to change `xs`.
   (f) All of the above.

7. If `(*v.begin() == *(v.end() - 1))` is true, which of the following must also be true about vector `v`?

   (a) `v.begin() == (v.end() - 1)`
   (b) `v.begin() != (v.end() - 1)`
   (c) `v.capacity() > 1`
   (d) `v.front() == v.back()`
   (e) `v.size() == 1`
   (f) `v.empty()`

8. Which of the following can be allocated dynamically?

   (a) `string`
   (b) `vector<int>`
   (c) `map<string, string>`
   (d) `int`
   (e) `double *`
   (f) All of the above

9. What does the following code output:

```
vector<int> xs = {1, 2, 3, 4};
for (auto x : xs) {
  xs.push_back(5);
}
cout << xs.size() << endl;
```

   (a) `4`
   (b) Syntax error
   (c) `5`
   (d) `9`
   (e) Undefined behavior

10. What does the following code output?

```
map<int, string> id_to_name = {
  {10, "Josh"},
  {20, "Emily"},
};
id_to_name.insert({30, "Abdol"});
string x = id_to_name[40];
cout << id_to_name.size() << endl;
```

   (a) None of the above (it won't compile)
   (b) 4
   (c) 3
   (d) 2

11. How many different vector objects (instances) are generated by the following code?

```
bool has_1(vector<int> input) {
  return find(
    input.begin(),
    input.end(),
    1) != input.end();
}

int main() {
  vector<int> v = {1, 2, 3};
  vector<int> v2 = v;
  cout << has_1(v) << endl;
}
```

   (a) 1 (c) 3 (e) 5
   (b) 2 (d) 4 (f) 6

12. What is wrong with the following code?

```
vector<int> v = {1, 2, 3};
while (v.size()) {
  cout << *v.begin() << endl;
  v.pop_back();
}
```

   (a) It will raise an exception because `push_back` will be called on an empty vector.
   (b) It will generate a syntax error because `*v.begin()` you can't dereference a vector.
   (c) It is an infinite loop because size can never be negative.
   (d) It will print `1` three times because it is popping from the end, not the beginning.

13. Which of the following iterators can be legally deferenced for a non-empty vector named `x`?

    (a) `x.begin()`
    (b) `x.cbegin()`
    (c) `x.rbegin()`
    (d) `x.crend()`
    (e) (a) and (b)
    (f) (a), (b) and (c)
    (g) All of the above.
    (h) None of the above.

14. Which of the following types provided by the STL are not templated?

    (a) `sstream`
    (b) `unordered_set`
    (c) `multimap`
    (d) `vector`
    (e) All of the above.
    (f) None of the above

15. In the lab where you created a SingleLink list, what are the cases that the `del` function member must account for?

    (a) If the argument is is the last element in the list.
    (b) If list is empty.
    (c) If the argument is is the first element in the list.
    (d) If the argument isn't present in the list.
    (e) If the argument is in the list (but isn't the first or last element).
    (f) All of the above.

16. In the lab where you created a SingleLink list, would a `out_of_range` exception need to be raised for a negative indices?

    (a) No, because no test case had negative indices.
    (b) No, because negative indices loop around like in Python.
    (c) Yes, because a negative index would result in a memory leak.
    (d) No, because negative indices are impossible.
    (e) Yes, because a negative index would access a memory position prior to the first element.
    (f) None of the above.

17. When should you call the `reserve` method of `vector`?

    (a) When you want to know the capacity of the vector
    (b) When you know the number of elements you intend to add to the vector
    (c) Before you destroy it, to ensure that all the elements are not leaked
    (d) Before each call to `push_back` to ensure there is space

18. A `vector`'s `crend()` returns what type of object?

    (a) An iterator pointing to the last element in the vector
    (b) An iterator pointing to the first element in the vector
    (c) An iterator pointing to one past the last element in the vector
    (d) An iterator pointing to one before the first element in the vector

19. Which of the following types can be elements in a `vector`?

    (a) `vector<string>`
    (b) `int *`
    (c) `map<string, int>`
    (d) (a) and (b)
    (e) (a) and (c)
    (f) (b) and (c)
    (g) (a), (b), and (c)
    (h) None of the above

20. What happens when you use the `[ ]` to access a key that doesn't exist in a `std::map`?

    (a) It inserts a default value for that key and returns it
    (b) It throws an exception
    (c) It returns false
    (d) It results in undefined behavior
    (e) None of the above

21. When will a container's begin and end iterators be equal?

    (a) When the container is empty
    (b) They will never be equal
    (c) When the container has exactly one element
    (d) When the container is const

22. In the `SingleLink` class of the last lab, why should a custom destructor be implemented?

    (a) Because the class has a custom default constructor
    (b) Because otherwise the `Node`'s within would be leaked.
    (c) Because the compiler would fail to work because the class has a pointer as a member attribute
    (d) Because the Rule of Three dictates that all classes must have a custom destructor
    (e) None of the above

23. In the `SingleLink` class of the last lab, calling the `append_back` function should cause which of the following to occur?

    (a) A `vector` to have increased in size by one
    (b) A call to `int`'s destructor
    (c) A `Node` to be dynamically allocated
    (d) A copy of the `SingleLink` to have have been created
    (e) None of the above

24. If a singly-linked list only had one data member (a pointer to a `Node` named `head_`), would it be possible to determine the size of such a list?

    (a) Yes, by using `head_->size()`.
    (b) Yes, by traversing each node with a loop, and incrementing a counter for each one until the end (`nullptr`) is found.
    (c) Yes, by using the `sizeof` function.
    (d) No, the list would require a second data member.

25. If a vector has the values {`'a', 'b', 'c'`}, what value must be added to the iterator returned by the `rend` member function to have it point at the `'b'` element?

    (a) Impossible to perform
    (b) -3
    (c) -2
    (d) -1
    (e) 0
    (f) 1
    (g) 2
    (h) 3

26. At what size will a vector's end iterator compare less than the begin iterator (i.e. `vec.end() < vec.begin()`)?

    (a) Never
    (b) 0
    (c) 1
    (d) Greater than 1

27. Which of the following function members of `vector` is often private (not exposed) in other languages (like Python and Java)?

    (a) Default constructor
    (b) `operator[]`
    (c) `capacity()`
    (d) `size()`

28. In the lab where you created a singly linked list, how did a Node refer to the next Node after it?

    (a) Using a non-const pointer.
    (b) Using a const reference.
    (c) Using a const pointer.
    (d) Using a non-const reference.
    (e) None of the above.

29. Which of the following `vector` member functions are non-const?

    (a) `empty`
    (b) `size`
    (c) `capacity`
    (d) `clear`

30. For a `vector` named `v`, if `v.front() == v.back()`, which of the following must be true?

    (a) v's begin and and end iterators must also be equal.
    (b) v must have a size of exactly 1.
    (c) v must have a capacity of 1.
    (d) v must have invoked the `push_back` member function.
    (e) v must be empty.
    (f) None of the above.

31. When you use a range-based for loop on a `map<string, int>`, what type is each element?

    (a) `map<string, int>`
    (b) `int`
    (c) `pair<const string, int>`
    (d) You can't use such a loop on a map.
    (e) `string`

32. What does the following code output?

```
map<string, string> name_to_city =
    {{"Josh", "EL"},
    {"Emily", "CL"}};
if (name_to_city["Mal"] == "DC")
    cout << "In DC ";
cout << name_to_city.size();
```

    (a) Undefined Behavior
    (b) `3`
    (c) `In DC 2`
    (d) `2`
    (e) Compiler Error

33. If the following line of code is legal, which of the following are possible types for `x`?

```
auto y = x->begin();
```

    (a) `vector<string> * x;`
    (b) `map<int, double> * x;`
    (c) `vector<int, double> x;`
    (d) `map<int, double> x;`
    (e) (a) and (b)
    (f) (c) and (d)
    (g) (a) and (c)
    (h) None of the above.

34. Which of the following operations are legal (i.e. will not cause an error or undefined behavior)?

```
vector<int> x {1, 2, 3};
auto y = x.cend();
```

    (a) `y[0];`
    (b) `--y;`
    (c) `*y;`
    (d) `y = 5;`
    (e) None of the above.

35. Which of the following is FALSE regarding `std::map`?

    (a) A map can have duplicate keys.
    (b) A map can have a pointer to it.
    (c) A maps value type can be another map.
    (d) A map can be const.
    (e) A map can have char values.
    (f) A map can have iterators to it.

This page intentionally left blank.