Week 10 Sample Exam

CSE 232 (Introduction to Programming II)

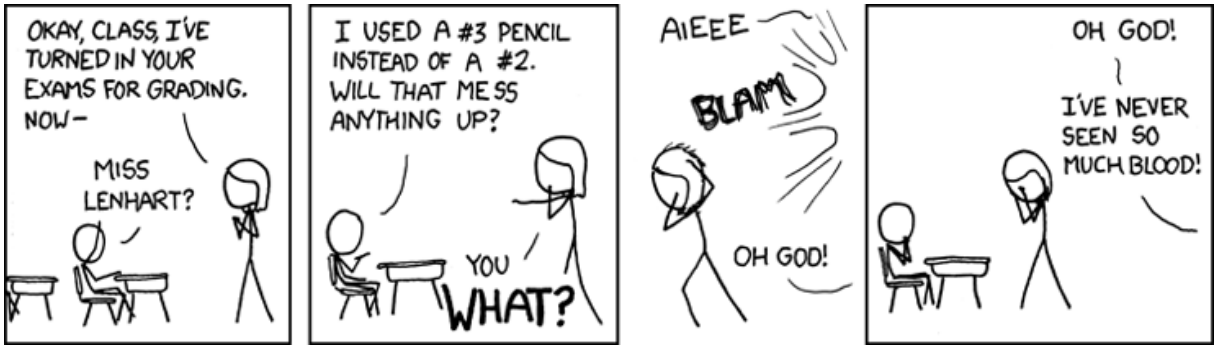
VERSION A

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 LETTER** (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). Please be sure to bring the required textbook!
- If you have any questions during the exam or when you finish the exam, please raise your hand and a proctor will attend you.



http://xkcd.com/499/ Date Accessed: October 16, 2024

1. What should be your default container to store elements in?

   (a) `vector`
   (b) `list`
   (c) `map`
   (d) `string`
   (e) `set`
   (f) `array`
   (g) `pair`

2. Which of the following member functions can change the amount of memory a vector is using?

   (a) `at`
   (b) `capacity`
   (c) `reserve`
   (d) `empty`
   (e) `size`
   (f) 2 of the above modify the amount of memory

3. In section 12.3, why would the second function named `get_number` fail to compile if `phone_book` was a `vector<Entry>` instead of a `list<Entry>`?

   (a) Iterators to `vector` can't be copied
   (b) `vector`'s iterators can't be incremented
   (c) `operator->` can't be used on `vector`'s iterators
   (d) `vector` doesn't permit access via range-based for loops
   (e) `operator!=` can't be used on `vector`'s iterators
   (f) It would compile correctly

4. The SingleLink class that was implemented in Lab 10 is structurally the same as which data structure?

   (a) `string`
   (b) `vector`
   (c) `map`
   (d) `forward_list`
   (e) `list`
   (f) `set`

5. How many elements are in `vec` after the following code executes?

```
string s{"abcd"};
vector<char> vec{s.begin(), s.end()};
vector<char> x = vec;
```

   (a) Impossible to determine
   (b) 4
   (c) 3
   (d) 2
   (e) 1
   (f) 0

6. Which of the following is guaranteed to have the same value as the following expression, assuming vec is a vector of ints?

```
vec.end() - vec.begin()
```

   (a) `vec.capacity()`
   (b) `vec.front()`
   (c) `vec.distance()`
   (d) `vec.reserve()`
   (e) `vec.size()`
   (f) `vec.back()`

7. If `x` is a `vector<int *>`, which of the following expressions will have type `int`?

   (a) `&x[0]`
   (b) `x++`
   (c) `x->begin()`
   (d) `*(x.at(0))`
   (e) `(x[0])->back()`
   (f) `x.size()`

8. Why should you not `push_back` on a vector that you are iterating over with a range-based for loop?

   (a) `push_back` will raise an exception if used in such a loop
   (b) Changing the size of a vector invalidates its iterators
   (c) Non-const member functions can't be used on const objects
   (d) Elements returned in a range-based loop are copied, not references
   (e) Vectors are immutable while being looped over
   (f) There's nothing wrong with using `push_back`

9. If the following expression compiles, what type must `x` be?
```
(x.at(0) == 'c')
```

   (a) `vector<pair<int, char>>`
   (b) `map<string, char>`
   (c) `vector<int>`
   (d) `vector<string>`
   (e) `string`
   (f) Impossible to determine with the information given

10. What is the output from the following code?
```
vector<int> vec = {1, 2, 3, 4};
vec += 1;
cout << vec.back();
```

   (a) `1`
   (b) Nothing is output
   (c) `12341`
   (d) It won't compile due to errors
   (e) `4`
   (f) Impossible to say due to undefined behavior
   (g) `5`

11. What does the following code output?
```
string s{"abca"};
map<char, int> m;
for (char c : s)
cout << m[c];
cout << "Size:" << m.size();
```

   (a) `0000Size:3`
   (b) `1111Size:4`
   (c) `abcaSize:3`
   (d) `0001Size:3`
   (e) `1111Size:3`
   (f) `1111Size:0`
   (g) `0000Size:0`
   (h) None of the above, the code wouldn't compile due to indentation errors

12. What is the output of the following code?
```
vector<int> input = {1, 2, 3, 4};
int & value = input.at(1);
value++;
for (auto elem : input) {
  cout << elem << " ";
}
```

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

13. What does the following code print?

```
for(int row{0}; row<5; ++row){
  for(int col{0}; col<5; ++col){
    std::cout << (row*col)%10
        << " ";
  }
  std::cout << "\n";
}
```

(a)
```
0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12
0 4 8 12 16
```

(b)
```
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

(c)
```
0 0 0 0 0 0
0 1 2 3 4 5
0 2 4 6 8 0
0 3 6 9 2 5
0 4 8 2 6 0
0 5 0 5 0 5
```

(d)
```
0 0 0 0 0
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
```

(e)
```
0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 2
0 4 8 2 6
```

14. 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) `1`
(b) `2`
(c) `23`
(d) `234`
(e) `1234`
(f) `123`

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

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

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

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

17. 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.

18. 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)

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

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

21. Which of the following can be allocated dynamically?

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

22. 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) Syntax error
    (b) Undefined behavior
    (c) 9
    (d) 4
    (e) 5

23. 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) 2
    (b) 3
    (c) 4
    (d) None of the above (it won't compile)

24. 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

25. What does the following code do?

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

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

26. 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.

27. 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

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

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

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

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

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

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

31. 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

32. 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

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

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

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

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

35. In the `SingleLink` class from lab, why should a custom destructor be implemented?

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

36. In the `SingleLink` class from 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 copy of the `SingleLink` to have have been created
    (d) A `Node` to be dynamically allocated
    (e) None of the above

37. 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.

38. 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

39. 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

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

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

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

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

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

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

43. 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 invoked the `push_back` member function.
    (c) v must have a size of exactly 1.
    (d) v must be empty.
    (e) v must have a capacity of 1.
    (f) None of the above.

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

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

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

46. 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.

47. 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.

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

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