$Id: cmps109-2015q3-exam1.mm,v 1.42 2015-07-06 12:02:38-07 - - $

page 1	page 2	page 3	page 4	page 5	Total / 54

Please print clearly :

Name :

Login : @ucsc.edu

Code only in C++11. No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Do your scratch work elsewhere and enter only your final answer into the spaces provided.

For all answers, assume : `using namespace std;`

1. Using an ***iterator***, write a function `sum` which finds the sum of a `vector<double>`. ***Do not use subscripts.*** **[2✔]**

```
double sum (const vector<double>& vec) {
```

2. Write the function `from_string` with a `string` argument. It returns an `int` if the string is valid and throws a `domain_error` if not. **[3✔]**

```
int from_string (const string &arg) {
   stringstream stream (arg);
```

3. Assuming the existence of `bool foo::operator< (const foo&)`, code `operator<=`. Do not assume the existence of any other operators. Specifically, ***do not*** use `operator==`. **[1✔]**

```
bool operator<= (const foo& left, const foo& right) {
```

4. Given a pair of iterators bounding a range, search that range for the first item that is equal to the third argument to the function. Return the iterator pointing at it. Return the end iterator if not found. **[2✔]**

```
template <typename Itor, typename Item>
Itor find (Itor begin, Itor end, const Item& item) {
```

5. Assuming `using value = pair<string,int>;`, write a function to print out all of the key and value pairs, one per line. Keys are `string`s. Each line of output should look like : `key = stringval:33`. That is, on each line, print the key, then an equal sign, then the string part of the value, a colon, and the number. Everything is held in a `map`. Use `auto` to declare the type of the map iterator, which must be a `const` reference. **[2✔]**

```
void print (const map<string,value>& m) {
```

6. Given a `list<int>` write a function that will return an iterator pointing at the largest element of the list. If the list is empty, return the end iterator. Note that a list has no `operator[]`. **[2✔]**

```
list<int>::iterator find_largest (const list<int>& ints) {
```

7. Write a function which accepts a vector and a list and uses `push_back` to append all of the positive integers (0 is neither positive nor negative) in the list onto the end of the vector. **[2✔]**

```
void (vector<int>& vec, const list<int>& lis) {
```

8. Write code that might appear at the beginning of `main` which loads all of the command line arguments (but not the execname) into `vector<string> args`. (Deduct 1 point if a loop is used.) **[2✔]**

```
int main (int argc, char** argv) {
```

9. Write code to copy an input vector into an output vector such that the output vector has elements in the reverse order. The input vector becomes empty after the copy. Do not use iterators. Do not use subscripts. Hint: `back`, `pop_back`, `push_back`. **[2✔]**

```
void copy_reverse (vector<string>& input, vector<string>& output) {
```

10. Write code to clone (make a complete recursive copy of) a tree. Assume the definition at the left. Make use of the constructor shown, and assume it has already been written. **[2✔]**

```
struct tree {
   int value;
   tree* left;
   tree* right;
   tree (int value,
      tree* left = nullptr,
      tree* right = nullptr);
};
```

```
tree* clone_tree (tree* t) {
```

11. Assuming class `ubigint` contains the following declarations,

```
using udigit_t = unsigned char;
using ubigvalue_t = vector<udigit_t>;
ubigvalue_t ubig_value;
```

code the unsigned addition operator. **[5✔]**

```
ubigint ubigint::operator+ (const ubigint& that) const {
```

12. Assuming class `intvec` partially shown here :

```
struct intvec {
   size_t size_;
   int* data_;
   intvec& operator= (const intvec&);
   explicit intvec (size_t size);
};
```

The size field indicates the number of elements in the data array. The data array might be `nullptr`, but only if the size field is 0.

(a) Code `operator=` as it would appear in the implementation file. **[3✔]**

```
intvec& intvec::operator= (const intvec& that) {
```

(b) Code the explicit `intvec` constructor, which allocates an array given by the size argument and fills it with zeros. **[2✔]**

```
explicit intvec::intvec (size_t size) {
```

Multiple choice. To the ***left*** of each question, write the letter that indicates your answer. Write ***Z*** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✔]**

number of correct answers		× 1 =	$= a$
number of wrong answers		× ½ =	$= b$
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	12		$= c$

1. What is the amortized speed of access to an element of a `map`?
   (A) $O(1)$
   (B) $O(\log n)$
   (C) $O(n)$
   (D) $O(n \log n)$

2. What statement will usually follow a statement like `#include <iostream>`?
   (A) `import static std namespace;`
   (B) `int main (char** argv, int argc);`
   (C) `using iostream = std;`
   (D) `using namespace std;`

3. A post-mortem report of memory leaks can be produced by what program?
   (A) `cpplint.py`
   (B) `g++`
   (C) `gdb`
   (D) `valgrind`

4. What is the portable way to declare a variable which will hold the result of calling the `sizeof` operator?
   (A) `int`
   (B) `long`
   (C) `size_t`
   (D) `unsigned`

5. which of the following is incorrect inside a class specification?
   (A) `friend: int i;`
   (B) `private: int i;`
   (C) `protected: int i;`
   (D) `public: int i;`

6. Inside the function `f`, how can one refer to the variable `p` in the following call: `p->f(x,y);`?
   (A) `auto`
   (B) `p`
   (C) `self`
   (D) `this`

7. Which operator can ***not*** be overloaded with either one or two operands, as appropriate, in the definition of the operator?
   (A) `operator*`
   (B) `operator+`
   (C) `operator++`
   (D) `operator<<`

8. Which is the appropriate way to catch an exception?
   (A) `catch (runtime_error error&)`
   (B) `catch (runtime_error error)`
   (C) `catch (runtime_error& error)`
   (D) `catch (runtime_error* error)`

9. Given the following four members of class `foo`, which will be called by the second of these two statements:
   `foo y; foo x = y;`?
   (A) `foo& operator=(const foo&);`
   (B) `foo();`
   (C) `foo (const foo&);`
   (D) `~foo();`

10. What is the equivalent to `a = *p++`, assuming `int a` and `int* p`?
    (A) `++p; a = *p;`
    (B) `a = *p; ++p;`
    (C) `a = ++p; *a;`
    (D) `a = p++; *p;`

11. The type `vector<string>` indicates that:
    (A) class `string` inherits from class `vector`.
    (B) class `vector` inherits from class `string`.
    (C) is the same as the type `char**`.
    (D) `vector` is a template class containing objects of class `string`.

12. Reference counting will fail to function without special handling on which data structure?
    (A) balanced binary search tree
    (B) directed acyclic graph
    (C) directed cyclic graph
    (D) linear linked list

Multiple choice. To the ***left*** of each question, write the letter that indicates your answer. Write ***Z*** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✔]**

number of correct answers		× 1 =	$= a$
number of wrong answers		× ½ =	$= b$
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	12		$= c$

1. What is the amortized time efficiency of `vector::push_back`?
   (A) $O(1)$
   (B) $O(\log n)$
   (C) $O(n)$
   (D) $O(n \log n)$

2. An `unordered_map` is implemented as a:
   (A) balanced binary search tree
   (B) graph of basic blocks
   (C) hash table
   (D) linear linked list

3. The function `f` uses, but does not modify its string argument. Which is its most appropriate declaration?
   (A) `void f (const string&);`
   (B) `void f (const string);`
   (C) `void f (string&);`
   (D) `void f (string);`

4. How does one declare a pointer to a string?
   (A) `string& p;`
   (B) `string* p;`
   (C) `string-> p;`
   (D) `string<auto> p;`

5. The syntax to portably print an end of line marker, independent of the operating system, is:
   (A) `cout << '\n';`
   (B) `cout << '\r';`
   (C) `cout << '\r\n';`
   (D) `cout << endl;`

6. In C++11 and C++14, the most appropriate way to refer to the null pointer is:
   (A) `0`
   (B) `NULL`
   (C) `null`
   (D) `nullptr`

7. The stream `cout` is associated with which standard file descriptor?
   (A) 0
   (B) 1
   (C) 2
   (D) 3

8. Which operator uses lazy (short-circuit) evaluation?
   (A) `--`
   (B) `<<`
   (C) `>>`
   (D) `||`

9. Which statement is appropriate to skip the rest of the statements inside a loop and restart the loop at the beginning?
   (A) `break`
   (B) `continue`
   (C) `goto`
   (D) `return`

10. Given the following `for`-statement, which is the correct way to test completion of iteration over an arbitrary collection `c`? (Fill in the blank.)
    `for (i = c.begin(); _____; ++i) f(*i);`
    (A) `i != c.end()`
    (B) `i < c.end()`
    (C) `i == c.end()`
    (D) `i > c.end()`

11. If the default constructor for class `foo` would normally not be implicitly generated, what statement inside class `foo` would force instantiation?
    (A) `foo() = 0;`
    (B) `foo() = default;`
    (C) `foo() = delete;`
    (D) `foo() = explicit;`

12. Inside the header file defining class `foo`, how is the destructor declared?
    (A) `!foo();`
    (B) `*foo();`
    (C) `?foo();`
    (D) `~foo();`