$Id: cmps109-2014q3-exam1.mm,v 1.39 2014-07-16 17:48:46-07 - - $

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

Please print clearly:

Name:	
Login:	@ucsc.edu

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.

1. Load the command line arguments (but not the program name) into a `vector<string>`. Just show the necessary code after the opening line of `main`. **[1✔]**

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

2. For class `foo`, write the prototypes for the six members that are automatically synthesized by the compiler. **[3✔]**

```
class foo {
```

3. Write the function `sum` that would return the sum of all elements in a range, given a pair of iterators that indicate the range in the usual way. Assume a dereferenced iterator returns a `double`. **[2✔]**

```
template <typename Iterator>
double sum (Iterator begin, Iterator end) {
```

4. Define a template class `stack`, which contains (does not inherit) a private `vector`. It supports the operations `push`, `pop`, `top`, `size`, and `empty`. Code all functions inline, using bodies which just call the appropriate `vector` function. Show everything that should be in the header file `stack.h` so that no implementation file `stack.cpp` is needed. Be consistent with the rest of the library. **[4✔]**

5. Define a single constructor for class `complex` which takes zero, one, or two arguments. For zero arguments, both fields are initialized to 0.0. For a single argument, the real field is initialized to its value, and the imaginary field to 0.0. For two arguments, the real field's value is the first argument, and the imaginary field's value is the second argument. Define the function entirely in the header. **[1✔]**

```
class complex:
   private:
      double real;
      double imag;
   public:
      complex (
```

6. Write a function that finds the smallest entry in a range given by a pair of iterators. `operator<` is used to determine what is smallest. If the range is empty, return the end of the range. If there are several smallest value, return the one nearest the start of the range. **[2✔]**

```
using Itor = vector<string>::const_iterator;
Itor smallest (Itor begin, Itor end) {
```

7. Write a `copy` function. It takes a pair of iterators delimiting a input range, and an object with a `push_back` function. It copies values from the input range and pushes each item thus found onto the back of the output object. **[3✔]**

```
template <typename Iterator, typename Container>
void copy (Iterator begin, Iterator end, Container& cont) {
```

8. Using the colon (foreach) form of a `for`-loop, print out the key and value pairs of a map, one key/value pair per line. **[2✔]**

```
map<string,int> si_map;

for (                                 : si_map) {
```

9. Write a loop which loads a map from `argv`, using all of the arguments except the name of the program, such that for all `i`, `argv[i]` is a key with the value `i`. **[2✔]**

```
int main (int argc, char** argv) {
   map<string,int> si_map;
```

10. Write a function `trim` which takes a `vector<char>` and removes all high-order zeros from the end of the vector. That is, it pops zeros from the end of the vector until the last element of the vector is non-zero, or the vector is empty. Note that the value 0 is being popped, not `'0'`. **[2✔]**

```
void trim (vector<char> vec) {
```

11. Write a function `split`, which takes a vector as an argument and splits the vector into two alternating output vectors with the even-numbered elements in the first vector and the odd-numbered elements in the second vector. For example, `split ({1,3,5,7,9,11,15,20,30,55})` will return `{1,5,9,15,30}` as the first of the pair and `{3,7,11,20,55}` as the second pair. In other words there are two resulting vectors, each containing alternating elements from the input vector. **[4✔]**

```
using intvec = vector<int>;
using vecpair = pair<intvec,intvec>;
vecpair split (const intvec& vec) {
```

12. Assuming the declarations presented here, write a function `execute` which takes an input line string: **[4✔]**
    (i) Using `split`, splits it into a vector of words, assuming that `split` splits on white space.
    (ii) The first word is the function name, which is found in a map.
    (iii) The function is then called with a pair of iterators, the first of which points at the second word in the vector, and the second of which is the end of the vector.
    (iv) Throw the exception `bad_command` either if there are no words on the command line or if the command is not found in the map.

```
using itor = vector<string>::const_iterator;
using function = void (*) (itor begin, itor end);
class bad_command: public exception {};
vector<string> split (const string&);
unordered_map<string,function> fnmap;
void execute (const string& command) {
```

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, which count 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. The area of memory used for dynamic memory allocation is the :
   (A) `function call stack`
   (B) `heap`
   (C) `static data`
   (D) `text area`

2. The copy constructor for class `foo`, defined in a header file, will look like :
   (A) `foo (const foo&&);`
   (B) `foo (const foo&);`
   (C) `foo (foo&&);`
   (D) `foo (foo&);`

3. Which of the following declarations will produce a warning ?
   (A) `int a (3.6);` // Note : parentheses.
   (B) `int b = 3.6;`
   (C) `int c {3.6};` // Note : braces.
   (D) `int d {};` // Note : braces.

4. In the following statements, what is the type of `n` ?
   ```
   vector<int> vi;
   auto n = vi.begin();
   ```
   (A) `vector::iterator`
   (B) `vector::iterator<int>`
   (C) `vector<int>::iterator`
   (D) `vector<int>::reverse_iterator`

5. An ancestor to C++ which used classes was :
   (A) Algol
   (B) Fortran
   (C) Lisp
   (D) Simula

6. Which one of these may ***not*** be used as an operator in C++ ?
   (A) `operator()`
   (B) `operator--`
   (C) `operator<>`
   (D) `operator[]`

7. To prevent the compiler from synthesizing the constructor with no arguments for class `foo`, use the definition :
   (A) `foo() = default;`
   (B) `foo() = delete;`
   (C) `foo() override;`
   (D) `virtual foo();`

8. What class performs memory management by using reference counting ?
   (A) `auto_ptr<T>`
   (B) `shared_ptr<T>`
   (C) `unique_ptr<T>`
   (D) None of the above. Memory management is done by garbage collection.

9. Storage menagement by reference counting fails on what kind of data structure ?
   (A) binary tree
   (B) cyclic graph
   (C) hash table
   (D) vectors and arrays

10. In the following, what statement is executed immediately after `continue` ?
    ```
    for (i = 0; i < n; ++i) {
       f(); continue; g();
    } h();
    ```
    (A) `++i`
    (B) `g()`
    (C) `h()`
    (D) `i<n`

11. The `operator=` assignment operator should always return its object by reference using the code :
    (A) `return *auto;`
    (B) `return *self;`
    (C) `return *that;`
    (D) `return *this;`

12. The keyword `explicit` :
    (A) Prevents default members, such as the default copy constructor, from being automatically provided.
    (B) Prevents implicit access to the standard input, output, and error streams.
    (C) Prevents local variables from being destroyed when they go out of scope.
    (D) Prevents one-argument constructors from behaving as automatic type conversions.

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, which count 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. Which container guarantees that all elements are stored in a contiguous block of memory?
   (A) `deque`
   (B) `list`
   (C) `map`
   (D) `vector`

2. Which expression is equivalent to `p+i`, given that `p` is a pointer and `i` is `ssize_t`?
   (A) `!p[i]`
   (B) `&p[i]`
   (C) `*p[i]`
   (D) `~p[i]`

3. Which operator uses lazy evaluation?
   (A) `++`
   (B) `<<`
   (C) `==`
   (D) `||`

4. What is the general efficiency of `vector::push_back`?
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(n \log_2 n)$

5. If `v.size() == n`, then `v.end()` usually means:
   (A) `&v[0]`
   (B) `&v[n+1]`
   (C) `&v[n-1]`
   (D) `&v[n]`

6. What statement should free memory that has been allocated with:
   `foo* p = new foo[n];`
   (A) `delete p;`
   (B) `delete* p;`
   (C) `delete[] p;`
   (D) `foo::~foo (p);`

7. If `m` is a `map`, then inside the loop `for(auto i:m)` we refer to the value associated with a given key by:
   (A) `i->first`
   (B) `i->second`
   (C) `i.first`
   (D) `i.second`

8. What statement will print one of the strings in a vector, given the following declarations?
   `vector<string> v;`
   `auto i = vec.cbegin();`
   (A) `cout << &i;`
   (B) `cout << **i;`
   (C) `cout << *i;`
   (D) `cout << i;`

9. An `unordered_map` is a:
   (A) balanced binary search tree
   (B) bidirectional linked list
   (C) hash table
   (D) lexicographically sorted array

10. What declaration would normally be expected to allow printing values of class `foo`?
    (A) `ostream& operator<< (const ostream&, const foo&);`
    (B) `ostream& operator<< (const ostream&, foo&);`
    (C) `ostream& operator<< (ostream&, const foo&);`
    (D) `ostream& operator<< (ostream&, foo&);`

11. The copy constructor and operator= will have to be explicitly written if a class contains what kind of field?
    (A) copiable object
    (B) pointer
    (C) primitive
    (D) reference

12. Which definition defines the ***prefix*** `operator++` (i.e., `++x`) when it is a member (not friend) of a class?
    (A) `foo operator++ (foo&, int);`
    (B) `foo operator++ (int);`
    (C) `foo& operator++ ();`
    (D) `foo& operator++ (foo&);`