$Id: cmps109-2013q1-exam2.mm,v 1.25 2013-02-27 17:30:13-08 - - $

page 1	page 2	page 3	Total / 31

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. Assume a declaration `intvector v;` followed by some code to put things into the vector. Finish the classes `intvector` and `intvector::iterator` adding only those members which are needed to make the following code work : `for (intvector::iterator i = v.begin(); i != v.end(); ++i) cout << *i << endl;`
   Code all functions as inline functions, not in the usual way as separate prototypes and implementations.

   (a) Code added directory to `intvector` : `begin` and `end`. **[4✓]**

```
class intvector {
   private:
      size_t size;
      int *data;
   public:
      class iterator;
```

   (b) Code added to `intvector::iterator` : Any function or operator used in the for-loop and any constructor needed by `begin` or `end`. Assume that `iterator` is declared inside of class `intvector`. **[6✓]**

```
class iterator {
   friend class intvector;
   private:
      int *pointer;



   public:
```

2. Define a template function called `printthem` which takes a pair of input iterators as arguments and prints each element of the data structure, one per line, assuming that `operator<<` is defined on the elements. **[1✓]**

```
template <typename itor>
void printthem (const itor &begin, const itor &end) {
```

3. Write a template function `copyreverse` whose argument is any vector passed in by constant reference and whose result, returned by value is a new vector. It uses a ***reverse*** iterator (`rbegin`, `rend`) to access successive elements of the argument vector. **[3✓]**

4. Consider an object-oriented hierarchy with a class `base`, from which is extended classes `rectangle` and `circle`. Code only those specific members/functions specified here and ignore the others.

   (a) Define an abstract base class `base` with a default constructor. Its only protected field is a serial number which is initialized successively to integers starting from 1, and an abstract function `area` which returns a `double`. **[2✓]**

   (b) Define a derived class `circle` which has a ctor that accepts a diameter as an argument and which overrides `area`. **[2✓]**

   (c) Define a derived clas `rectangle` whose ctor takes a length and a width (both `double`s) and implements `area`. **[2✓]**

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. **[11✔]**

number of correct answers		× 1 =	= a
number of wrong answers		× ½ =	= b
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	11		= c

1. What form of polymorphism describes template classes and functions ?
   (A) conversion
   (B) generic
   (C) inheritance
   (D) overloading

2. If one wishes to define a stack of integers in terms of a vector, but prohibit all vector operations from being used by the stack's client, except for those explicitly listed, what is the proper form of inheritance ?
   (A) `class stack: private vector<int> {`
   (B) `class stack: protected vector<int> {`
   (C) `class stack: public vector<int> {`
   (D) `class stack: template vector<int> {`

3. If we declare `foo x;` and call a function with the syntax `f(x);` what declaration of `f` will allow it to modify `x` ?
   (A) `void f (const foo &);`
   (B) `void f (foo &);`
   (C) `void f (foo *);`
   (D) `void f (foo);`

4. If a module's interface is specified in `foo.h`, what should the first non-comment line be ?
   (A) `#define __FOO_H__`
   (B) `#ifdef __FOO_H__`
   (C) `#ifndef __FOO_H__`
   (D) `#include __FOO_H__`

5. It is necessary to make a destructor virtual if any :
   (A) constructor is virtual
   (B) member field is a pointer
   (C) member function is virtual
   (D) time inheritance is used

6. If class `complex` is implemented as a pair of `doubles`, what is an appropriate overloaded constructor ?
   (A) `~complex (vector<double>);`
   (B) `~complex (double, double);`
   (C) `complex (double r = 0, double i = 0);`
   (D) `explicit complex (double r);`

7. If it is desirable to suppress creation of `operator=` in a given class, define it as a member and append what to the end of the definition in the header ?
   (A) `= default`
   (B) `= delete`
   (C) `= virtual`
   (D) `= void`

8. The average speed of `vector::push_back` is :
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(n \log_2 n)$

9. Given `foo *p;` the expression `++p` changes the address in `p` by how many bytes ?
   (A) `sizeof(1)`
   (B) `sizeof(*p)`
   (C) `sizeof(foo)`
   (D) `sizeof(uintptr_t)`

10. The `operator--` is available on what kind of iterator ?
    (A) bidirectional
    (B) forward
    (C) input
    (D) output

11. By default, members of a `class` are __(x)__ and members of a `struct` are __(y)__.
    (A) (x) = private, and (y) = private
    (B) (x) = private, and (y) = public
    (C) (x) = public, and (y) = private
    (D) (x) = public, and (y) = public