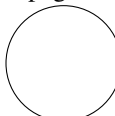
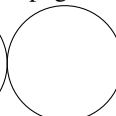
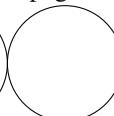
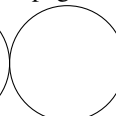



\$Id: cmpls109-2019q1-midterm.mm,v 1.164 2019-02-08 15:12:30-08 - - \$

page 1	page 2	page 3	page 4	Total / 42
				

PLEASE PRINT CLEARLY :

NAME :

CRUZID :

@ucsc.edu

No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Points will be deducted for messy or unreadable answers. Do your scratch work elsewhere and enter only your final answer into the spaces provided.

- Write a complete program, everything that needs to be typed into the source file **hello.cpp** in order to print the following message to the standard output. **[1✓]**

```
-bash-36$ ./hello
Hello, world!
```

- Assuming **iterator i, j**, write prototypes for the member operators prefix **++**, binary **!=**, and unary *****. as they are used in the following examples : **++i, i!=j, and *i.** **[3✓]**

```
class iterator {
public:
```

- Assuming a header file containing the following (shown at right), show the code as it appears in the implementation file for the following two functions. Both functions update **uvalue** in place and neither calls any other function.

(a) **divide_by_2** **[3✓]**

```
class ubigint {
private:
    vector<uint8_t> uvalue;
    void divide_by_2();
    void multiply_by_2();
    ...
```

(b) **multiply_by_2** **[3✓]**

4. Rewrite the following statement using the two-semicolon version of the **for**-loop, explicitly using the iterators provided by the container. Your statement should have semantics identical to this statement. [1✓]

```
for (auto elem&: cont) foo (elem);
```

5. Write a main function that reads in words from the standard input and keeps a count of the number of times each word appears. At end of file, print out each word, in lexicographic order, one per line, followed by the number of times it appears. Use the declaration **string word;**. The loop **while (cin >> word)** will read in one word and stop at end of file. Use an appropriate container. Example output is shown. Do not show any **#include** statements, just the main function. [2✓]

```
bar 1298
foo 32
hello 9
widget 3
```

6. Write a **copy** function which takes forward iterators pointing into an input container, and a forward iterator pointing at an output container. Copy the items from the input container to the output container. It is not possible to verify whether or not the output container has enough space, so just assume that it has. [2✓]

```
template <typename in_itor, typename out_itor>
void copy (in_itor in_begin, in_itor in_end, out_itor out_begin) {
```

7. Define a template class **stack**. All methods are to be declared inline. Do not show any of the implicitly declared methods, since, for this class, they are all acceptable for the purposes of this class.
- (a) Use a private field of class **vector** to hold the stack. [1✓]
 - (b) Function **pop** removes the top element of the stack but does not return it. [1✓]
 - (c) Function **top** returns the top element of the stack by reference, but does not alter the stack. [1✓]
 - (d) Function **push** enters a new element onto the stack. [1✓]
 - (e) Function **empty** indicates whether or not the stack is empty. [1✓]

8. Write the prototypes for class **foo** which will be implicitly generated unless otherwise specified. Show them as they would appear in a header file inside the class definition. Show prototypes only. Point allocation is given in the table at left. [2✓]

6 correct: 2	✓	class foo {
5 correct: 1½	✓	public:
4 correct: 1	✓	
3 correct: ½	✓	
else: 0	✓	

9. Complete the following operators, assuming **operator==** and **operator<** are defined. [2✓]

```
template <class T>
inline bool operator!= (const T& x, const T& y) {
```

```
template <class T>
inline bool operator> (const T& x, const T& y) {
```

```
template <class T>
inline bool operator<= (const T& x, const T& y) {
```

```
template <class T>
inline bool operator>= (const T& x, const T& y) {
```

10. Code a linear search **find** which takes two iterators and a comparison function which returns true if its two arguments are equal. Make the usual assumptions. [2✓]

```
template <typename iterator, class comparator>
iterator find (iterator begin, iterator end, comparator equal) {
```

11. Code **find_if** which takes two iterators and a predicate and returns an iterator pointing at the first element for which the predicate is true. (A predicate is a function which returns a **bool** result.) [2✓]

```
template <typename iterator, class predicate>
iterator find_if (iterator begin, iterator end, predicate pred) {
```

12. Code **find_min** which returns an iterator pointing at the minimum element in a range. Its function argument **less** returns true if the first argument is less than the second argument. [2✓]

```
template <typename iterator, class less_fn>
iterator find_min (iterator begin, iterator end, less_fn less) {
```

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		$\times 1 =$	$= a$
number of wrong answers		$\times \frac{1}{2} =$	$= b$
number of missing answers		$\times 0 =$	0
column total $c = \max(a - b, 0)$	12		$= c$

- For a container **c** which provides random access iterators, what expression returns the same result as **c.size()** ?
 (A) **c.begin() - c.end()**
 (B) **c.end() - c.begin()**
 (C) **c.end() - c.begin() + 1**
 (D) **c.end() - c.begin() - 1**
- Which container allocates all the contents of the data structure in a single block of storage on the heap ?
 (A) **deque**
 (B) **list**
 (C) **map**
 (D) **vector**
- What will redirect both stdout and stderr to the same file ?
 (A) **prog 1>foo.out 1>&2**
 (B) **prog 1>foo.out 2>&1**
 (C) **prog 2>>1 1>foo.out**
 (D) **prog | foo.out 2>&1**
- Given an iterator pointing at an arbitrary position within a container, which container will allow an insertion at that position with a worst case time of $O(1)$?
 (A) **deque**
 (B) **list**
 (C) **string**
 (D) **vector**
- What does this declaration do ?
class A { friend class B; };
 (A) Allows **B** to inherit virtual functions from **A**, but only if they are **protected**.
 (B) Grants **A** access to the private parts of **B**, and also grants **B** access to the private parts of **A**.
 (C) Grants **A** access to the private parts of **B**, but not vice-versa.
 (D) Grants **B** access to the private parts of **A**, but not vice-versa.
- Which is a copy constructor of class **foo** ?
 (A) **foo (const foo &&);**
 (B) **foo (const foo &);**
 (C) **foo (foo &&);**
 (D) **foo (foo &);**
- What statement usually follows library **#includes** at the start of an implementation file ?
 (A) **#include <std::>**
 (B) **import std package;**
 (C) **using namespace std;**
 (D) **using package std;**
- Which **#include** is needed to compile the statement **cout<<3; ?**
 (A) **<cstdio>**
 (B) **<iomanip>**
 (C) **<iostream>**
 (D) **<stdio.h>**
- A destructor must be virtual if any ____ is virtual.
 (A) constructor
 (B) member data field
 (C) member function
 (D) nonmember function
- Which statement should appear in the following member function ?
foo& foo::operator= (const foo& that) {
 (A) **if (&this == *that)**
 (B) **if (*this == that)**
 (C) **if (this == &that)**
 (D) **if (this == that)**
- Given an arbitrary iterator **i**, what is the most efficient way of incrementing it ?
 (A) **++i**
 (B) **i++**
 (C) **i+=1**
 (D) **i=i+1**
- The default copy constructor is almost certainly wrong if one of the members of the class is :
 (A) a function
 (B) a pointer
 (C) a primitive
 (D) an object of some class

