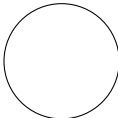
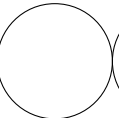
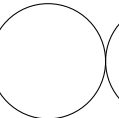
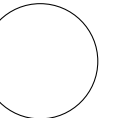
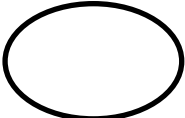


\$Id: cmps109-2019q3-midterm.mm,v 1.106 2019-07-22 16:25:18-07 - - \$

page 1	page 2	page 3	page 4	Total / 42	PLEASE PRINT CLEARLY :				
					<table border="1"> <tr> <td>Name :</td> <td></td> </tr> <tr> <td>CruzID :</td> <td>@ucsc.edu</td> </tr> </table>	Name :		CruzID :	@ucsc.edu
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.

1. Finish the implementation of the function `minimum`, which returns an iterator pointing at the minimum element in the range. Assume that the items pointed at by the iterators have `operator<` available. [2✓]

```
template <typename iterator>
iterator minimum (iterator begin, iterator end) {
```

2. Finish the implementation of the function `find`, which returns true if some element in the range is equal to the third argument. Assume that the items pointed at by the iterators have `operator==` available. [2✓]

```
template <typename iterator, typename item_t>
bool find (iterator begin, iterator end, item_t item) {
```

3. Write a non-template function called `sum`. The argument is a `vector<double>` and the result is a `double`. [2✓]

4. Write a template function called `tsum`, whose template argument is an arbitrary type of iterator, and whose function arguments are a pair of iterators indicating a range. Its result is a `double`. Assume the iterators point at doubles. [2✓]

5. Implement `ubigint::trim`, a member function which ensures that a `ubigint` is in canonical form. Canonical form means that there are no leading zeros and that zero is represented by an empty vector. The name of the vector field is `ubig_value`. [2✓]

6. Assume that `bigint::operator<` has been implemented. Code a non-template non-member `operator>` whose arguments are two `bigints`. [1✓]

7. Assume class `foo` has a prefix member `operator++` defined for it. Define the postfix `operator++` as a non-member which makes use of `foo::operator++()`. Assume `foo` has a copy ctor. [2✓]

8. Circle.

- (a) Write the function `circle`, which returns, as a `pair`, the area (as the first field) and circumference (as the second field) of an circle, given its radius. Use `M_PI` from `<cmath>`. All numbers are `double`. For the mathematically unsophisticated, $A = \pi r^2$ and $C = 2\pi r$. [1✓]

- (b) Given the declaration `auto x = circle (e);` using `x`, write one line of code to print out the area and circumference. The line of output (assuming appropriate data) should look like the following: [1✓]
area = 23.2134, circumference = 17.0795

9. 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. Assume the output iterator is such that the output container does not overflow or cause memory errors. [2✓]

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

10. Define `equal_range` which takes two pairs of forward iterators and returns true if all of the elements of the ranges are equal and the ranges are of the same length. There is no `size()` function, and forward iterators can not be subtracted. Assume the elements of the range have `operator==`. [3✓]

```
template <typename itor1, typename itor2>
bool equal_range (itor1 begin1, itor1 end1, itor2 begin2, itor2 end2) {
```

11. Assuming `string s;` and `string t;` write an expression (not a complete statement) equivalent to `s == t`, but use only `operator<` and no other comparison operators. Use some combination of the operators `and (&&)`, `or (||)`, and `not (!)`. Use parentheses as appropriate. [1✓]

12. Code a fragment of a **Makefile** which will build an object (`.o`) file from a C++ (`.cpp`) source file. Show the wildcard target, prerequisite, and command. Assume the following macro has already been defined. [1✓]

```
COMPILECPP = g++ -std=gnu++17 -g -O0 -Wall -Wextra -Wold-style-cast
```

13. Code a non-member `multiply_by_2` function which is useable by `ubigint` which will double the value represented by a vector argument. Write code as a loop manipulating each digit individually. Do not call any functions from `multiply_by_2`. The value is to be updated in place. [3✓]

```
void multiply_by_2 (vector<unsigned char>& value) {
```

14. Define the function `print`. It prints out all of the elements in a range assuming that `operator<<` is defined for the elements in the range. A single space is printed out between successive elements of the range, but no space is printed before the first or after the last element. [2✓]

```
template <typename iterator>
void print (iterator begin, iterator end) {
```

15. The Collatz conjecture states that for any arbitrary positive integer n , if n is even, replace n by $n/2$, but if n is odd, replace n by $3n + 1$. Repeat until $n = 1$, and the procedure will terminate. Write the non-template function `collatz`. Its argument is of type `int` and it returns a `vector<int>` containing a trace of a sequence of collatz numbers. The resulting vector will contain all of the numbers in sequence, starting from the original argument, and ending with the number 1. Here is a sample program that calls `collatz`, followed by its output. [3✓]

```
int main() {
    vector<int> a {30, 5, 11};
    for (int i: a) {
        vector<int> v = collatz (i);
        for (int j: v) cout << " " << j;
        cout << endl;
    }
}
-bash-$ ./collatz
30 15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
5 16 8 4 2 1
11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

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		$= c$

- Multiplication of an m -digit number by an n -digit number takes how many single-digit multiplications?
 - $O(m * n)$
 - $O(m + n)$
 - $O(m - n)$
 - $O(m / n)$
- How should the function **f** be declared so that it accepts a **string** by constant reference?
 - `void f (const string&);`
 - `void f (const string);`
 - `void f (const string*);`
 - `void f (const string->);`
- Memory mangement using **shared_ptr** needs special handline for what kind of data structure?
 - graph possibly with cycles
 - graph with no cycles
 - linear non-circular linked list
 - vector of pointers to objects
- It is expected that **v.size()** should return a value equivalent to:
 - `v.begin() - v.end()`
 - `v.begin() - v.end() + 1`
 - `v.end() - v.begin()`
 - `v.end() - v.begin() + 1`
- What keyword will allow another arbitrary class or function access to all members of the class?
 - friend**
 - private**
 - protected**
 - public**
- What keyword will allow access only to members of the class, but also to any class derived from it?
 - friend**
 - private**
 - protected**
 - public**
- What will produce an object (**.o**) file but suppress linking to an executable binary?
 - `g++ -MM`
 - `g++ -c`
 - `g++ -g`
 - `g++ -o`
- How long does it take to insert, delete, or find elements of a **map**?
 - $O(1)$
 - $O(\log n)$
 - $O(n)$
 - $O(n \log n)$
- How long does it take to insert, delete, or find elements of an **unordered_map**?
 - $O(1)$
 - $O(\log n)$
 - $O(n)$
 - $O(n \log n)$
- What is the correct (and preferred) way to link down a list?
 - `while (p != 0)`
 - `while (p != NULL)`
 - `while (p != null)`
 - `while (p != nullptr)`
- Given the expression **p->f(what)**, inside the code for the function **f**, what expression is used to refer to the value of **p**?
 - self**
 - that**
 - this**
 - what**
- When an object belonging to a class goes out of scope, what is automatically called?
 - the constructor
 - the destructor
 - the garbage collector
 - the **free** function

GRADE INFLATION

