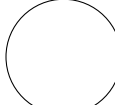
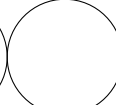
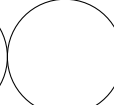
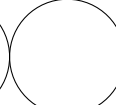



\$Id: cse111-2019q4-midterm.mm,v 1.132 2019-11-01 15:39:56-07 - - \$

page 1	page 2	page 3	page 4	Total / 42
				

Last Name :	
First 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, and ONLY in the spaces provided.*

- Write a complete C++ program : everything that needs to be in the file **echo.cpp**, which implements the **echo(1)** command. It prints out each of its command line arguments with exactly one space in between arguments, but no space before the first argument or after the last argument. **[4✓]**

```
-bash-1$ echo hello world
hello world
-bash-2$ echo

-bash-3$ echo foo bar baz qux
foo bar baz qux
```

- Complete the function **trim** which will ensure a **ubigint** is in canonical form. Assume a field declared as **vector<udigit\_t> ubigvalue;** **[1✓]**  
**void ubigint::trim() {**
- Code a template function **print** which prints to **cout** all of the elements of a container, one element per line. Assume **operator<<** is defined for the elements. The function should work on any container, such as **list**, **vector**, etc. It has a single argument, which is a container.
  - Code **print** using the colon version of a **for**-loop. Make only implicit, not explicit, use of an iterator. **[2✓]**
  - Code **print** using the three-part version of a **for**-loop using the container's **cbegin** and **cend** functions. Assume the container has only forward iterators (not bidirectional, not random access). **[3✓]**

4. Code the function `equal_range`, whose template arguments are two types of forward iterators and a comparison function type, and whose function arguments are two begin and end ranges and an equal comparison function object. It returns true if all elements in the ranges are equal and both ranges have the same number of elements. [3✓]

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

5. Given the outline of `bigint` shown here, implement `bigint::operator<` as it appears outside of the class definition. Use `ubigint::operator<` for the necessary comparison, but no other `ubigint` operator or function. [2✓]

```
class bigint {
private:
    bool is_negative;
    ubigint uvalue;
public:
    bool operator< (const bigint& that) const;
};
```

6. Write the function `ubigint::divide_by_2` which divides by 2 the value of a `ubigint` as per the project specifications. Update the value in place and do not call any other members of `ubigint`. [2✓]

<pre>class ubigint { private:     vector&lt;unsigned char&gt; value; public:     void divide_by_2() { };</pre>	<pre>void ubigint::divide_by_2() {</pre>
--	--

7. Write a function `dup_adjacent` which searches a range and returns true if any two adjacent (next to each other) elements are equivalent, based on the comparison parameter. For example, the sequence {1,2,3,3,4,5} returns true, while {8,3,7,5,9,3} returns false, because the two 3s are not adjacent. The function must run in  $O(n)$  time. [3✓]

```
template <typename itor_t, typename equal_t>
bool dup_adjacent (itor_t begin, itor_t end, equal_t equal) {
```

8. Write a function `minimum_itor` whose argument is a constant reference to a `vector<double>` and which returns a `vector<double>::iterator` pointing at the minimum element in the vector. If there are more than one element equal to the minimum, point the iterator at the element closest to the beginning of the range. Make an appropriate assumption if the vector is empty. [2✓]

9. There are six automatically generated class members. The default constructor and the destructor are given here for class `foo`. List the prototypes (signatures) of the others as they would appear in a class header file.

<pre>class foo { public:</pre>	(a) The copiers. [1✓]
<pre>    foo();     ~foo();</pre>	(b) The movers. [1✓]

10. Given the definition of `tstack` as shown at the left, show the implementation of the class as it would appear in outside of the class definition itself. (Remember your data structures course?)

```
template <typename item_t>
class tstack {
private:
    struct node;
    using node_ptr =
        shared_ptr<node>;
    struct node {
        item_t value;
        node_ptr link;
    };
    node_ptr top_ptr;
public:
    bool empty() const {
        return top_ptr == nullptr;
    }
    const item_t& top() const;
    void pop();
    void push (const item_t&);
};
```

```
template <typename item_t>
const item_t& tstack<item_t>::top() const { // (a) [1✓]
}

template <typename item_t>
void tstack<item_t>::pop() { // (b) [1✓]
}

template <typename item_t>
void tstack<item_t>::push (const item_t& n) { // (c) [2✓]
}

}
```

11. Write a program that will copy the standard input to the standard output. Use `getline` for input. [2✓]

```
#include <iostream>
#include <string>
using namespace std;
int main() {
```

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		$= c$

- What term is used to describe a virtual function in a derived class that has the same signature as a function of the same name in its base class?
  - delete
  - inherit
  - overload
  - override
- What is the best parameter description for a function that sums a vector, but does not change any of its elements? (Fill in the blank.)  
`double sum (____);`
  - `const vector<double>`
  - `const vector<double>&`
  - `vector<double>`
  - `vector<double>&`
- Which data structure will require special handling if managed by `shared_ptrs`, in order to avoid memory leak?
  - arbitrary  $n$ -way tree
  - balanced binary search tree
  - cyclic graph
  - linear linked list
- Which data structure should be used to maintain a table whose operations are insert, remove, and find by key, without any need to iterate in any particular order, in order to minimize  $O(f(n))$  time for these operations?
  - `list<pair<string, string>>`
  - `map<string, string>`
  - `unordered_map<string, string>`
  - `vector<pair<string, string>>`
- What usually follows `#include <iostream>` so that it is possible to use the name `cout` instead of always using `std::cout`?
  - `#include <std::packages>`
  - `import namespace std;`
  - `typedef std::namespace;`
  - `using namespace std;`
- When it is necessary to print an object of a particular class via `operator<<`, what keyword is generally used in the declaration of `operator<<`?
  - `friend`
  - `private`
  - `protected`
  - `public`
- Given classes `circle` and `square`, both of which inherit from abstract class `shape`, how should `area` be defined in the class definition of `shape`, knowing that it requires a concrete object in order to compute the area?
  - `virtual double area() = 0;`
  - `virtual double area() = default;`
  - `virtual double area() = delete;`
  - `virtual double area() {return nullptr;}`
- What is the link step in a `Makefile`?
  - `$(EXEC) : $(OBJS)`  
`$(GPP) -o $(EXEC) $(OBJS)`
  - `%.o : %.cpp`  
`$(GPP) -c $<`
  - `$(OBJS) : $(CPPSRC)`  
`$(GPP) -o $(OBJS) $(CPPSRC)`
  - `$(EXEC) : $(CPPSRC)`  
`$(GPP) -c $@`
- Given that `i` is an object of some iterator class, what is assumed to be the most efficient way to increment it in the third part of a `for`-loop?
  - `++i`
  - `i++`
  - `i+=1`
  - `i=i+1`
- Given the declaration `foo x;`, and assuming that `foo` has a non-static field called `n`, in the expression `x.f()`, how would the function `f` refer to the field `n`?
  - `foo::n`
  - `this->n`
  - `this.n`
  - `this::n`
- Which of the following must be declared in every C++ program?
  - `int main (char**, int);`
  - `int main (int, char**);`
  - `int main (string&);`
  - `int main (vector<string>&);`
- What should be the first non-comment line in the file `foo.h`?
  - `#define __FOO_H__`
  - `#ifdef __FOO_H__`
  - `#ifndef __FOO_H__`
  - `#include __FOO_H__`