$Id: cmps109-2019q1-final.mm,v 1.103 2019-03-21 10:32:48-07 - - $

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

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.

1. Define a template **operator<** which takes two **pair**s by constant reference and returns true if the first is lexicographically less than the second. Definition : $(a, b) < (c, d)$ if $a < c$ or if $a = c$ and $b < d$. Do not use any comparison operator other than **operator<**. Assume that **operator<** is defined on both the first and second parts of the pair. The two pairs may be of different types, requiring only that **operator<** is defined on their respective elements. **[2✔]**

2. Given the outline of a definition of **bstset**, which implements a set as a balanced binary search tree, show the implementation of each of the following. Much of the definition has been omitted for the sake of brevity.
   (a) Implement **bstset::contains**, as it would appear outside of the class definition. Do not assume any operators other then the parameter **less**. **[4✔]**
   (b) Implement **bstset::~bstset** as an inline function (inside the class definition). **[1✔]**
   (c) Implement **bstset::node::~node** as it would appear outside of the class definition. **[3✔]**

```
template <typename item_t, typename less_t = std::less<item_t>>
class bstset {
   private:
      struct node {
         item_t item {};
         node* left {nullptr};
         node* right {nullptr};
         ~node();
      };
      node* root {nullptr};
      less_t less;
   public:
      bool contains (const item_t& key);
      ~bstset()
};
```

3. Define **`differentiate`** which performs symbolic differentiation on a polynomial. Represent the polynomial with the exponent as the subscript and the coefficient as the value. To differentiate, for each term in the sum of the form kx^n, the resulting derivative term is knx^{n-1}. **[3✔]**

   Example 1 : $\frac{d}{dx}\ ax^3 + bx^2 + cx + d \ = \ 3ax^2 + 2bx + c$

   Example 2 : $v = 5x^3 + 9x^2 + 8x + 10$ is represented as **`polynomial v {10, 8, 9, 5}`**

   Its derivative $d = 15x^2 + 18x + 8$ is represented as **`polynomial d {8, 18, 15};`**

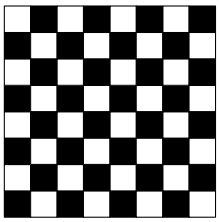
```
using polynomial = vector<double>;
polynomial differentiate (const polynomial& p) {
```

4. Define the template function **`copy_if`**. It has three template paramaters : an input iterator, an output iterator, and a unary predicate. It has four function parameters : begin and end input iterators indicating a range, an output iterator, and a unary predicate function or function object. It copies all elements from the input to the output for which the predicate returns true. Assume the output iterator accesses a container with sufficient space. **[2✔]**

5. Fill in the blanks : **[1✔]**

```
const GLubyte BLACK[] = _____________________________________________;
const GLubyte WHITE[] = _____________________________________________;
```

6. Write a function to draw a chess board, which is an 8×8 grid of alternating black and white squares with a white square at the lower right corner. The parameters **`xpos`** and **`ypos`** are the coördinates of the lower left corner of the chess board, and **`sqsize`** is the size of one of the squares. Use a loop nested within a loop. Use the definitions of **`WHITE`** and **`BLACK`** above. **[4✔]**



```
void draw_chessboard (GLfloat xpos, GLfloat ypos, GLfloat sqsize) {
```

7. Define `operator++` in both the prefix and postfix formats as they would appear as member functions in class `foo` as they would appear in the header file, assuming that the implementations will appear in some other file. **[1✔]**

```
class foo {
   public:
```

8. Define `merge`, which merges two input ranges into a single output range, given a pair of iterators for each of the input ranges, an iterator for the output range, and a less-than comparison function. Assume both input ranges are sorted into ascending order as specified by the `Less` argument. **[3✔]**
   Example call : `merge (a.begin(), a.end(), b.begin(), b.end(), back_inserter(v), less<int>());`

```
template <typename Initor1, typename Initor2,
          typename Outitor, typename Less>
void merge (Initor1 begin1, Initor1 end1, Initor2 begin2, Initor2 end2,
            Outitor result, Less less) {
```

9. Define classes `animal`, `cat`, and `dog`.
   (a) Class `animal` has a private constant `string` name field which is initialized by the constructor which takes a `string` as an argument. It has a virtual constant function `name`, which returns the name, and an abstract virtual constant function `noise`, which returns a string. **[3✔]**
   (b) Derived classes `cat` and `dog` override the `noise` function : For `dog`, return the string `"woof"`. For `cat` return the string `"meow"`. **[3✔]**

   Following is an example program using these classes.

```
int main() {
   vector<shared_ptr<animal>> va {
      make_shared<dog> ("argos"),
      make_shared<cat> ("bast"),
   };
   for (const auto& a: va) {
      cout << a->name() << " "
           << a->noise() << endl;
   }
   return 0;
}
```

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		× 1 =	$= a$
number of wrong answers		× ½ =	$= b$
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	12		$= c$

1. What is the most appropriate way to define **`MAX`** so that its value is guaranteed to be determined at compile time ?
   (A) **`#define MAX 10;`**
   (B) **`const size_t MAX = 10;`**
   (C) **`constexpr size_t MAX = 10;`**
   (D) **`final size_t MAX = 10;`**

2. How is class **`stack`** declared as a polymorphic class ?
   (A) **`class stack<T>`**
   (B) **`template <struct T> class stack`**
   (C) **`template <typename T> class stack`**
   (D) **`typename <class T> class stack`**

3. Given an iterator pointing into an arbitrary position within a container, which container will allow an $O(1)$ insertion at that point ?
   (A) **`deque`**
   (B) **`list`**
   (C) **`string`**
   (D) **`vector`**

4. If a **`map`** contains 1 000 000 items, what is the expected number of comparisons to be made by **`find`** ?
   (A) 1
   (B) 20
   (C) 1 000
   (D) 1 000 000

5. If a virtual function in a derived class has the same signature as a virtual function in its base class, then it should be declared with what attribute in the derived class ?
   (A) **`abstract`**
   (B) **`const`**
   (C) **`override`**
   (D) **`static`**

6. To find if an element is in a collection of elements, which container will provided the fastest search ?
   (A) **`list`**
   (B) **`set`**
   (C) **`unordered_set`**
   (D) **`vector`**

7. Given a **`map<foo,bar>`**, what operation must be provided to the **`find`** function for it to apply to a **`foo`** ?
   (A) **`operator()`**
   (B) **`operator<`**
   (C) **`operator==`**
   (D) **`struct hasher`**

8. What is equivalent to **`c.back()`** for a container that provides a direct (random) access iterator ?
   (A) **`*(c.end()-1)`**
   (B) **`*c.end()`**
   (C) **`*c.end()-1`**
   (D) **`c.end()`**

9. What is expected to be true for an empty container ?
   (A) **`c.begin()!=c.end()`**
   (B) **`c.begin()<=c.end()`**
   (C) **`c.begin()<c.end()`**
   (D) **`c.begin()==c.end()`**

10. What statement should be used to fill in the blank in the following function so that it compiles but will produce memory leak ?
    `void f() { ____ }`
    (A) **`auto p = delete string();`**
    (B) **`auto p = make_shared<string>();`**
    (C) **`auto p = make_unique<string>();`**
    (D) **`auto p = new string();`**

11. If **`0x12345678`** is sent from a big-endian host to a little-endian host without translation for network byte order, what value will be received ?
    (A) **`0x12345678`**
    (B) **`0x56781234`**
    (C) **`0x78563412`**
    (D) **`0x87654321`**

12. If the statement
    `auto i = m.find (key);`
    succeeds in searching a **`map`**, what expression will access the value that was wanted ?
    (A) **`i->first`**
    (B) **`i->second`**
    (C) **`i.first`**
    (D) **`i.second`**

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		× 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 provides the best locality of reference ?
   (A) **deque**
   (B) **list**
   (C) **unordered_map**
   (D) **vector**

2. Which is the destructor for class **foo** as declared outside the class ?
   (A) **~foo::~foo**
   (B) **~foo::foo**
   (C) **foo::~foo**
   (D) **foo::foo**

3. What is the size of the boundary tag overhead for each call to **new** ?
   (A) **2 * sizeof (char)**
   (B) **2 * sizeof (float)**
   (C) **2 * sizeof (int)**
   (D) **2 * sizeof (uintptr_t)**

4. How many siginficant bits are there in the **status** variable updated by **waitpid**(2) ?
   (A) 8
   (B) 16
   (C) 24
   (D) 32

5. What should be the intializer for
   const GLubyte DARK_RED[] _____;
   (A) **{ 0, 255, 255};**
   (B) **{127, 0, 0};**
   (C) **{255, 0, 0};**
   (D) **{255, 127, 127};**

6. When is the value of a **constexpr** determined ?
   (A) at compile time
   (B) at link time
   (C) when exec starts the program
   (D) when the function is called

7. What kind of cast is used to convert a **char*** to a **uintptr_t** ?
   (A) **const_cast**
   (B) **dynamic_cast**
   (C) **reinterpret_cast**
   (D) **static_cast**

8. In the declaration
   using xmap = map<string,double>;
   which type is the same as **double** ?
   (A) **xmap::iterator_type**
   (B) **xmap::key_type**
   (C) **xmap::mapped_type**
   (D) **xmap::value_type**

9. What should appear in the blank ?
   ```
   %.o: %.cpp
           ${GPP} -c ____
   ```
   (A) **$<**
   (B) **$>**
   (C) **$?**
   (D) **$@**

10. If an **unordered_map** contains 1 000 000 items, what is the expected number of comparisons to be made by **find** ?
    (A) 1
    (B) 20
    (C) 1 000
    (D) 1 000 000

11. What is the proper way to catch an exception ?
    (A) **catch (exception exn)**
    (B) **catch (exception& exn)**
    (C) **catch (exception* exn)**
    (D) **catch (exception-> exn)**

12. How many bits are there in an IPv6 address ?
    (A) 16
    (B) 32
    (C) 64
    (D) 128