$Id: cmps109-2016q1-exam2.mm,v 1.50 2016-02-18 17:26:51-08 - - $

page 1 page 2 page 3 Total / 32

Please print clearly :

Name : SOLUTION

Login : @ucsc.edu

Code only in C++11. 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. Math functions in a hash table.

   (a) Write a `using` statement that defines the type of a math function like `sqrt`, `log`, or `sin`. **[1✓]**

```
using math = double (*)(double);
```

   (b) Define a table called `math_hash` which uses an initializer list to initialize a hash table whose keys are `strings` and whose values are math functions of the same name. Initialize the table with the functions `sqrt`, `log`, and `sin`. **[2✓]**

```
unordered_map<string,math> math_hash {
      {"sqrt", sqrt},
      {"log", log},
      {"sin", sin},
};
```

   (c) Write a function `eval` which returns a `double` and has two arguments : a `string` representing the name of a function, and an argument to the function. If the function is found in the above table, its value when applied to the second argument is returned. If the function is not found, throw a `domain_error`. **[2✓]**

```
double eval(const string& fn, double arg){
   auto f = math_hash.find(fn);
   if (f == math_hash.end()) throw domain_error("eval");
   return f(arg);
}
```

2. Define a template function `copy_if`. It has three template parameters : the type of an input iterator, the type of an output iterator, the type of a predicate. It has four function arguments : begin and end input iterators for the source container, a begin iterator for the target container, and a predicate function. All elements for which the predicate is true are copied from the input container to the output container. **[3✓]**

```
template <typename initor, typename outitor, typename pred>
void copy_if(initor b, initor e, outitor b2, pred p){
   while (b != e) {
      if (p(b)) *b2++ = *b;
      ++b;
   }
};
```

3. Define a function `reverse`, which reverses elements in any container bounded by iterators and does not use any auxiliary memory. It makes use of `std::swap`. **[2✓]**

```
template <typename Iter>
void reverse (Iter begin, Iter end) {
   while (begin != end and begin != --end) {
      swap(*begin, *end);
      ++begin;
   }
}
```

4. Each of the boxes here represents one kind of polymorphism. In each, write ***universal*** or ***ad hoc*** to indicate which category. Also, write one of the following terms to indicate the more specific form: ***conversion***, ***overloading***, ***parametric***, ***inclusion***. **[2✓]**

`void foo (int);` `void foo (double);` ad hoc overloading	`void bar (double);` `bar (3);` ad hoc conversion
`class qux: public baz {` `};` universal inclusion	`template <typename T>` `T sum (T*);` universal parametric

5. Code a binary search. Given are two direct access iterators and a key to be located in the range bounded by the iterators. You may use any operator defined on direct access iterators. Assume that the range is sorted into increasing order by `operator<` on the elements in the range. Do not use any comparison operator other than `operator<` on the elements pointed at by the iterators. Return an iterator pointing at the element found, or the end iterator. **[4✓]**

```
template <typename itor, typename item>
itor binary_search (itor begin, itor end, const item& it) {
    itor ε = end;
    while (begin < ε) {
        itor m = begin + (ε - begin)/2;
        if (*m < it) begin = mid + 1;
        else if (it < *m) ε = mid;
        else return mid;
    }
    return end;
}
```

6. Code a linear search. Given are two input iterators and a key to be located. Use only operators that are permitted on input iterators. For comparing keys, use only `operator==`. Return the iterator pointing at the key found or the end iterator. If the key occurs more than once, return the first one found. **[2✓]**

```
template <typename itor, typename item>
itor linear_search (itor begin, itor end, const item& it) {
    for (; begin != end; ++begin) if (*begin == it) break;
    return begin;
}
```

7. As in the project, given

```
class ubigint {
   private:
      using udigit_t = unsigned char;
      using ubigvalue_t = vector<udigit_t>;
   public:
      void trim();
```

assume `vector <udigit_t> vec;`

write the function `trim` as it appears in `ubigint.cpp` which removes all high-order zeros from the vector. The number zero is represented as an empty vector. **[2✓]**

```
void ubigint::trim() {
    while (vec.size()>0 and vec.back()==0)
        vec.pop_back()
}
```

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 expression is the same as `&v[0]`, if `v` is a `vector`?
   B
   (A) `&&v.begin()`
   (B) `&*v.begin()`
   (C) `*&v.begin()`
   (D) `**v.begin()`

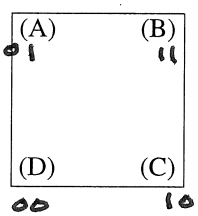
2. What kind of cast is used to convert a `char*` to a `long unsigned`?
   C
   (A) `const_cast`
   (B) `dynamic_cast`
   (C) `reinterpret_cast`
   (D) `static_cast`

3. `const GLubyte RED[]`
   C
   (A) `= {0x00, 0x00, 0xFF};`
   (B) `= {0x00, 0xFF, 0xFF};` R, G, B
   (C) `= {0xFF, 0x00, 0x00};`
   (D) `= {0xFF, 0xFF, 0x00};`

4. What is the running time of `map::operator[]`?
   B
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(n \log_2 n)$

5. Given `map<Key,Value> x;` what is the type of `x.begin()`?
   B
   (A) `const pair<Key,Value>`
   (B) `pair<const Key,Value>`
   (C) `pair<Key,Value>`
   (D) `Value`

6. Using the OpenGL coordinate system from the project, where is the point (0,0)? Assume that the wrong answers refer to points (0,1), (1,0), and (1,1).
   D
   (A) upper left
   (B) upper right
   (C) lower right
   (D) lower left



7. What happens with the following code?
   ```
   list<int>v {1,2,3,4,5};
   auto i = v.begin(); int j = 100;
   while (j-->0) ++i;
   ```
   D
   (A) Fails to compile.
   (B) Goes into an infinite loop.
   (C) Segmentation fault (core dumped)
   (D) Terminates normally.

8. For class `foo`, what is the move constructor?
   B
   (A) `foo (const foo&);`
   (B) `foo (foo&&);`
   (C) `foo& operator= (const foo&);`
   (D) `foo& operator= (foo&&);`

9. For class `foo`, what should the parameter list look like for the following definition?
   ```
   ostream& operator<< ( ____);
   ```
   C
   (A) `const ostream&, const foo&`
   (B) `const ostream&, foo&`
   (C) `ostream&, const foo&`
   (D) `ostream&, foo&`

10. If `char c` contains an ASCII digit, which statement will convert it into an equivalent binary number?
    C
    (A) `c = (int) c;`
    (B) `c = c + '0';`
    (C) `c = c - '0';`
    (D) `c = isdigit (c);`

11. In a `Makefile` what should be put in the blank in the following recipe?
    ```
    %.o : %.cpp
          ${COMPILECPP} -c ____
    ```
    ≠<
    C
    (A) `$$`
    (B) `$*`
    (C) `$<`
    (D) `$@`

12. In the `listmap` project, if there are n integers in a `listmap<int>`, how many pointers are there in the entire data structure?
    B
    (A) $2n$
    (B) $2n+2$
    (C) n
    (D) $n+1$