Answer Key to Midterm Exam 1

Fall 2014
Time: 50 minutes.
Fri, Sep 26

| Total points: 100 plus one extra credit question worth 5 points | 7 pages |

Please use the space provided for each question, and the back of the page if you need to. Please do not use any extra paper. The space given per question is a lot more than sufficient to answer the question. Please be brief. Longer answers do not get more points!

- No electronic devices of any kind. You can open your textbook and notes
- Please leave your UB ID card on the table
- This booklet must not be torn or mutilated in any way and must not be taken from the exam room
- Please stop writing when you are told to do so. We will not accept your submission otherwise.
- If you wanted to, you can answer the extra credit question without answering all of the other questions

---

Your name: _______________________________________

Your UBIT Name: __________________________________

---

The rest of this page is for official use only. Do not write on the page beyond this point.

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Score obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>name and id (5 max)</td>
<td></td>
</tr>
<tr>
<td>Problem 1 (30 max)</td>
<td></td>
</tr>
<tr>
<td>Problem 2 (30 max)</td>
<td></td>
</tr>
<tr>
<td>Problem 3 (20 max)</td>
<td></td>
</tr>
<tr>
<td>Problem 4 (15 max)</td>
<td></td>
</tr>
<tr>
<td>Extra credit problem (5 max)</td>
<td></td>
</tr>
<tr>
<td>Total Score: (105 max)</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (30 points). Mark the correct choice(s) or give a brief answer. Each question is worth 3 points. All codes are in C++.

1. Which of the following are declarations but not definitions? Check all that apply.
   - struct Foo;
   - string s;
   - string s("David");
   - string foo(string s);
   - typedef string my_string;
   - None of the above

2. Consider the following definition char var[] = "Hello\0World";
   What does cout << var; print?
   - Hello
   - World
   - Hello World
   - it prints nothing
   - it can’t even compile because var is not a string

3. Will the following program compile without error?  
   - YES
   - NO

   ```
   #include <iostream>
   using namespace std;
   int main() { cout << foo() << endl; return 0; } 
   int foo() { return 1; } 
   ```

4. Will the following program compile without error?  
   - YES
   - NO

   ```
   #include <iostream>
   using namespace std;
   int foo() { return 1; } 
   int main() { cout << foo() << endl; return 0; } 
   ```

5. Consider the following snippet of C++

   ```
   void ubswap(int*& a, int*& b) {
       int* temp = a; a = b; b = temp;
   }

   int main() {
       int x = 1, y = 9;
       int* u = &x; int* v = &y;
       int** a = &u; int** b = &v;
       ubswap(u, v);
       return 0;
   }
   ```

   which pairs of variables in main are swapped? Check all that apply.
   - x and y
   - u and v
   - a and b
6. Suppose you wanted to make use of Lexer routines I gave and all your codes are put in yourprog.cpp. The interface for the Lexer is declared in Lexer.h and the implementation is stored in Lexer.cpp, all in the same directory. The Lexer.h header is properly included. Which of the following compilation commands will produce Lexer.o? (Check all that apply.)

- g++ -c yourprog.cpp
- g++ yourprog.cpp Lexer.cpp
- g++ Lexer.cpp yourprog.cpp
- g++ Lexer.o yourprog.cpp
- g++ -c Lexer.cpp
- g++ yourprog.cpp Lexer.cpp -o best
- g++ Lexer.cpp -o yourprog.cpp

7. Continue with the previous question. Suppose we already have Lexer.o in the same directory, which of the following commands will produce an executable?

- g++ -c yourprog.cpp
- g++ yourprog.cpp Lexer.cpp
- g++ yourprog.cpp Lexer.cpp -o best
- g++ yourprog.cpp Lexer.o
- g++ yourprog.cpp Lexer.o -o best
- g++ -c Lexer.cpp
- g++ Lexer.cpp -o yourprog.cpp

8. Write a C++ line that defines a new type named mytype_t. The type is a function pointer to a function that takes two pointers to int and returns a string.

```
typedef string (*mytype_t)(int*, int*);
```

9. Let foo be a function with the prototype int foo(int); Suppose we want to define a variable var so that later we can assign var["abc"] = &foo; How would we define var?

```
// directly like this
map<string, int (*)(int)> var;
// OR, indirectly like this
typedef int (*mytype_t)(int);
map<string, mytype_t> var;
```

10. Consider the following helloWorld.cpp file

```
#include <iostream>
using namespace std;
int main() { cout << "Hello world" << endl; return 0; }
```

Write the content of a Makefile that when we type make will produce an executable named hw

```
all:
g++ helloWorld.cpp -o hw
```
Problem 2 (30 points). You can assume that using namespace std; and all appropriate #include statements have been written at the top of the file.

1. (15) Write a C++ function foo() that takes a vector myvec of integers by reference and reverse all elements of the vector.

```cpp
void foo(vector<int>& myvec) {
    for (int i=0; i<myvec.size()/2; ++i) {
        swap(myvec[i], myvec[myvec.size()-1-i]);
    }
}
```

2. (15) Write a C++ function bar() that takes a stack st of int as argument, and returns the bottom element of the stack. For example,

```cpp
int bar(stack<int> st) {
    while (st.size() > 1)
        st.pop();
    return (st.size() == 1 ? st.top() : 0);
}
```
Problem 3 (20 points). You can assume that `using namespace std;` is at the top of the file. Write a function `foo()` that does the following. It takes an vector of `int` named `vec` as input, and returns the maximum number of occurrences of an integer in the vector. For example,

```
vec = [ 3 9 -2 9 3 3 -5 3 ] ------> foo() returns 4
vec = [ 3 9 2 -1 5 4 -7 6 ] ------> foo() returns 1
```

In the former, number 3 occurs most frequently. In the latter, the number of occurrences of any number is 1, so the maximum is also 1. Of course, if `vec` is empty then 0 should be returned.

```cpp
int foo(vector<int> vec)
{
    int maxsofar = 0;
    int currentCount = 0;
    sort(vec.begin(), vec.end());
    for (int i=0; i<vec.size(); ++i) {
        currentCount++;
        if (i > 0 && vec[i] > vec[i-1]) {
            maxsofar = max(maxsofar, currentCount);
            currentCount = 0;
        }
    }
    return max(maxsofar, currentCount);
}
```
Problem 4 (15 points). You can assume that using namespace std; and all appropriate #include statements have been written at the top of the file. Consider the Token type in the Lexer.h file in assignment 3:

```cpp
enum token_types_t { IDENT, BLANK, TAG, ERRTOK, ENDTOK };
struct Token { token_types_t type; std::string value; };
```

Suppose we already tokenized an input HTML expression, found no invalid token and no unknown tag, and stored all tokens in a token vector vec. Write function valid() that takes vec as argument and returns whether the expression is well-formed. For example, suppose the input expression is

```html
<red>Hello world</red>
```

Then, vec is the following vector

```cpp
[ (TAG, "red"), (IDENT, "Hello"), (BLANK, " "), (IDENT, "world"), (TAG, "/red") ]
```

and valid() should return true. On the other hand, if the expression was

```html
<red>Hello<blue></red>
```

Then valid() should return false because vec is

```cpp
[ (TAG, "red"), (IDENT, "Hello"), (TAG, "blue"), (TAG, "/red") ]
```

```cpp
bool valid(vector<Token> vec)
{
    stack<string> st;
    for (int i=0; i<vec.size(); i++) {
        switch (vec[i].type) { 
            case TAG:
                if (vec[i].value[0] == '/') {
                    if (!st.empty() && st.top() == vec[i].value.substr(1)) {
                        st.pop();
                    } else {
                        return false; // unmatched closing TAG
                    }
                } else {
                    st.push(vec[i].value);
                }
            break;
            case IDENT:
            case BLANK:
            default:
                break; // skip over all those guys
        }
    }
    return st.empty();
}
```
**Problem 5** (5 points extra credit problem). Suppose we tokenized the input expression but didn’t put the tokens in a vector; instead, we pushed them all onto a stack (in the same scanning order from left to right of the input expression), and pass the stack of tokens to `valid()`. Describe in English how you would write `valid` now? (In particular, no code has to be written, just a couple of lines describing your idea is sufficient.)

Pop one token out at a time, treat open TAG as close TAG and vice versa. All else is the same as in the vector case.