Exam I Review Questions — KEY

Spring 2011

The following review questions are similar to the kinds of questions you will be expected to answer on Exam I (scheduled for Oct. 14), which will cover LCR, chs. 1–4, and TCS, chs. 1–4. **Make sure you do your best to answer each question before you look at this Key**; that will help you to learn. If you have any questions about the answers, please ask them in class or ask your Teaching Assistants in the review session. The actual exam will be about this length.

1. (4 points)

Please answer the following questions.

a. What `#include` statement do you put at the top of a program that does uses `cin` or `cout`?
`#include <iostream>`

b. What `using` statement do you always put at the top of your programs (to get standard naming conventions)?
`using namespace std;`

c. Declare an integer variable `N` and initialize it to 10.
`int N = 10;`

d. Give a statement to increase the value of variable `N` by 10.
`N = N + 10;`

e. What library would you include if you wanted to use the `sin`, `cos`, and `tan` functions?
`<cmath>`

f. What library would you include if you wanted to use pseudo-random numbers?
`<cstdlib>`

See LCR 4 (p. 77); TCS 3.3.

2. (5 points)

Complete the code segment below to print “`There are` *XXX* `days.`” if integer x has a value 1 through 7 inclusive. Replace *XXX* with the value of variable `x`; note the period.

```
int x;
cin >> x;
// add your code here
if ((1 <= x) && (x <= 7)) {
    cout << There are " << x << " days.\n";
}
```

See LCR 4, pp. 86–9.

3. (6 points)

What is the final value of each expression? Watch for integer division.

a. `12 + 6.8 / 2`

```
15.4
```

b. `15 / 6.0 + 2`

```
4.5
```

c. `(int)30.55 / 4;     // casting has precedence over division`

```
7
```

See TCS 2.8, 3.1; LCR 3, pp. 49–50.

4. (4 points)

You have an integer variable `x` that holds a four-digit positive integer. Write C++ code to store the middle two digits in integer variable `y`.

Ans: [TCS 4.1]

`y = x / 10 % 100;` **-OR-** `y = x % 1000 / 10;`

5. (10 points)

What is the output from each code segment below?

a.
```
int x = 1, y = 0;
  if (x > 0 && y < 0) {
      x = y = 23;
  }
  cout << x << "  " << y << endl;
```
```
1  0
```

b.
```
int x = 1, y = 0;
  if (x > 0 || y < 0) {
      x = y = 23;
  }
  cout << x << "  " << y << endl;
```
```
23  23
```

c.
```
x = 10;  y = 40;
 if (x >= 10) {
     if (y < 40) {
         y++;
     } else {
         y--;
     }
     cout << x << "  " << y << endl;
```

```
10  39
```

d.
```
x = 10;  y = 40;
if (x >= 10) {
    if (y < 40) {
       y++;
    }
} else {
    y--;
}
cout << x << "  " << y << endl;
```
```
10  40
```
See TCS 4.2–4.5; LCR 4, pp. 86–9.

e. In part d above, under what condition(s) would `y` be incremented?

Ans: If x is greater than or equal to 10 and y is less than 40.

6. (6 points)

How many times does each loop execute?

a.
```
for (i = 0; i <= 100; i++) {
    robot.translate(1);
}
```
101 times

b.
```
for (i = 1; i < 100; i++) {
    robot.translate(1);
}
```
99 times

See LCR 3, pp. 60–3.

7. (6 points)

Examine the code below. What (not how) does the following function do?

```
int mystery(int x, int y, int z)
{
  int temp;    // ultimately, holds returned value

  temp = x;    // assume the largest is x
  if(y > temp) { temp = y; }
  if(z > temp) { temp = z; }
  return temp;

}
```

Ans: It returns the maximum of x, y, and z.

8. (8 points)

The Fibonacci numbers are defined:

fibonacci (0) = 1
fibonacci (1) = 1
fibonacci (*n*) = fibonacci (*n*–1) + fibonacci (*n*–2);

Write a *recursive* C++ function to compute the Fibonacci numbers:
Ans: [TCS 4.7–4.9]

```
int fibonacci (int n) {
     // fill in
     if ((n == 0) || (n == 1)) {
          return 1;
     } else {
          return fibonacci(n-1) + fibonacci(n-2);
     }
}
```

9. (4 points)

What is printed in each case:

a)
```
cout <<  3  *  4+5  << endl;
```
```
17
```

b)
```
cout << 17 + 10 / 4 << endl;
```
```
19
```

c)
```
string N = "17";
cout << N + "10 / 4" << endl;
```
```
1710 / 4
```
[LCR 3, pp. 50-1]

d)
```
int N = 17;
cout << (N = 16) << endl;
```
```
16
```

10. (6 points)

Suppose that every time `currentTime()` is called, it returns the current time in the form of an integer (typically the number of seconds since a certain day in 1970). Using `currentTime()` and `motors()`, write C++ code to make the robot move forward at half speed for 45 seconds.
Ans: [LCR 4, pp. 84–5]

```
     int start = currentTime();
```

```
while ((currentTime() - start < 45) {
     robot.motors(0.5,0.5);
}
```

11. (4 points)

If `A` and `B` are boolean variables, then the expression `(!A) || (!B)` is equivalent to which of the following conditions:

a) `(! (A && B))`
b) `(! (A || B))`
c) `(!A) && (!B)`
d) `true`

Ans: (a) [LCR 4, p. 89]

12. (9 points)

Study the following code segment.

```
int N = 7;
int i, x;

for (i = 1; i <= N; i++)  {
    cin >> x;
    cout << i << ". " << x << "    ";
    if(i % 3 == 0) { cout << endl; }
}
cout << endl;
```

a. What is displayed with the given input? Watch the `endl` characters.
Input: `11    12    13    14    15    16    17    18    19`

```
1. 11    2. 12    3. 13
4. 14    5. 15    6. 16
7. 17
```

b. In terms of `N`, what is the value of the loop control variable upon termination?

```
N+1
```

13. (4 points)

1. Rewrite the following C++ statement to make it more robust (less prone to error).

```
if (x / y != 0 && y != 0) { cout << x / y << endl; }
```

`Answer: The operands of && are evaluated in order (not mentioned`

previously, but now you know!), and we don't want to divide by 0, so:
```
     if (y != 0 && x / y != 0) { cout << x / y << endl; }
```

2. Rewrite the following C++ code segment to make it more efficient and to remove any unnecessary code.

```
            if (x < y && y > x) { cout << "Not good.\n"; }
            cout << "Carry on.\n";
```

Answer: "x < y" and "y > X" always have the same truth value, so:
```
     if (x < y) { cout << "Not good.\n"; }
     cout << "Carry on.\n";
```

3. Rewrite the following C++ code segment to make it more efficient and to remove any unnecessary code.

```
            if (x < y && y >= x) { cout << "Not good.\n"; }
            cout << "Carry on.\n";
```

Answer: Since x<y implies x<=y, which is equivalent to y>=x:
```
      if (x < y) { cout << "Not good.\n"; }
      cout << "Carry on.\n";
```

4. Rewrite the following C++ code segment to make it more efficient and to remove any unnecessary code.

```
            if (x < y || y >= x) { cout << "Not good.\n"; }
            cout << "Carry on.\n";
```

Answer: Similar explanation to previous.
```
      if (y >= x) { cout << "Not good.\n"; }
      cout << "Carry on.\n";
```

14. (5 points)

Explain the difference between an *interoceptors* and an *exteroceptors*? Give an example of each.

Ans: Interoceptors are sensors used for sensing the internal state of the robot; exteroceptors are external sensors for sensing the environment of the robot. Examples of interoceptors include `getStall()` and `getBattery()`. Examples of exteroceptors include IR proximity detectors, light detectors, and video cameras. See LCR 4, p. 72.