CS 102 / ECE 206 — Spring 2011

Review Questions II — KEY

The following review questions are similar to the kinds of questions you will be expected to answer on Exam II (April 7), which will focus on LCR, chs. 5–7 and TCS, chs. 5–13. **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. The actual exam will be about 15 questions.

1. (5 points)

Write a C++ `main()` function that reads `double` numbers from a file called "`in.dat`" until it reaches the end-of-file, and prints the numbers on `cout` left-justified in a field of width 6 with three digits after the decimal point. Your program should not use vectors or arrays (which it does not need). Make sure to show any required `#include`s.

Ans: [LCS pp. 123–6, 184–9]

```
#include <iostream>
#include <fstream>
int main () {
     ifstream infile ("in.dat");
     while (!infile.eof()) {
          double datum;
          infile >> datum;
          printf ("%-6.3f\n", datum);
     }
     infile.close();     // not required, but good form
     return 0;
}
```

2. (5 points)

Define an enumeration types `Suit` and `Rank` for suits of cards (`CLUBS`, `DIAMONDS`, `HEARTS`, `SPADES`) and their ranks (`ACE`, `TWO`, …, `KING`). The `Rank` enumeration should be defined so that `ACE` has the numerical value 1 and the others follow consecutively (`TWO` = 2, etc.).

Ans:

```
enum Suit { CLUBS, DIAMONDS, HEARTS, SPADES };
enum Rank { ACE=1, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT,
            NINE, TEN, JACK, QUEEN, KING };
```

3. (5 points)

State in your own words the difference between *reactive* (or *direct*) *control* and *behavior-based control*.

Ans: See LCR, p. 171.

4. (5 points)

For the following questions, tell (1) what the function does with the argument, (2) what can be done to the argument within the function, and (3) how the original argument is affected by the action of the function.

(a) When an argument is passed by **value** to a function [e.g., `f(int v)`], what happens?

Ans: (1) The function copies the actual parameter; (2) the value can be read, but assignments to the formal parameter do not affect the actual parameter; (3) the actual parameter cannot be changed by the function.

(b) When a r**eference variable** is passed to a function [e.g., `f(int& v)`], what happens?

Ans: (1) The function receives a reference to (address of) the actual parameter, which is usually more efficient than copying it; (2) assignments can be made to the formal parameter, which (3) change the corresponding actual parameter.

(c) When a **constant reference** variable is passed to a function [e.g., `f(const int& v)`], what happens?

Ans: (1) The function receives a reference to (address of) the actual parameter, which is usually more efficient than copying it; (2) assignments cannot be made to the formal, and therefore (3) the actual cannot be changed.

[See TCS 8.6, 8.7, 9.5]

5. (5 points)

Complete function `read_vec(N)` to read values into an integer vector of size `N` and return it. Stop when either the vector is full or you reach the end of the file. You may assume valid data.

```
vector<int> read_vec (const int N) {
```

//Ans [LCR pp. 123–6]:

```
    vector<int> v(N);
    for(int i = 0; i < N && !cin.eof(); i++)
       cin >> v[i];
```

//end of ans.

```
    return v;
}
```

6. (5 points)

You have the following code fragment. What is printed with the indicated input values?

```
int x;
cin >> x;
switch(x)  {
    case 40:  x %= 6;
    case 60:  x--;  x--;  break;
    case 80:  x += 586;
    default:  if(x != 0)  x = -x;
}
cout << x << endl;
```

a) input: 40

Ans: 2

b) input: 80

Ans: -666

7. (5 points)

Given the following structure definitions, write an output command to print (as an integer) the `suit` of the last card in `deck`. (Note that decks need not contain 52 cards.)

```
struct Card { int suit, rank; };
struct Deck {
  vector<Card> cards;
};
Deck deck;
// deck initialization would be here
// your answer:
```

Ans:

```
cout << deck.cards[deck.cards.size()-1].suit << endl;
// endl is optional
```

8. (5 points)

You have vectors `U` and `V` declared as:

```
vector <int> U (10); // create vector with 10 elements, all 0
vector <int> V (10, -1); // create vector with 10 elem., all -1
```

Write code to copy `V` to `U`. Hint: One line of code is all that is needed.

Ans: `U = V;`

9. (5 points)

Suppose that `Card` and `Deck` are defined as in question #7, and suppose that you are given a `bool` function `greater(c1, c2)` that tests if card `c1` is greater than card `c2`. Complete the following definition of a `bool` function `ordered(d)` that returns `true` if deck `d` is sorted in ascending order and `false` otherwise. (Note that `d` may have less than 52 cards.)

```
     bool ordered (const Deck& d) {
```

//Ans:

```
       for (int i=0; i < d.cards.size()-1; i++) {
         if (greater (d.cards[i], d.cards[i+1])) return false;
       }
       return true;
```

//end of ans.

```
     }
```

10. (5 points)

State the *Information Hiding Principles*.

Ans: The implementer should have all the information needed to implement a module correctly, <u>and nothing more</u>. The user of a module should have all the information needed to use a module correctly, <u>and nothing more</u>. (To get complete credit, you need to mention both the implementer and the user (or client) and "nothing more" [the hiding part].) [Discussed in class]

11. (5 points)

Describe in words and show in a diagram (you must do both to get full credit) how you would wire up a Braitenberg vehicle so that it exhibits the following behavior:

The vehicle accelerates towards light directly in front of it, but tends to veer away from light that is on one side or the other of it.

Ans: The left sensor is connected to the left motor and the right sensor to the right motor. See diagram for "Coward" in *LCR*, p. 145.

12. (5 points)

Fill in the new array values after the code segment is executed. Leave no blank array elements.

```
const int N = 8;
int i, A[N] = {0, 11, 22, 33, 44, 55, 66, 77};

for (i = 1; i < N-1; i++)       // look carefully!
  A[i] = A[i+1];
```

```
                 _________________________________________
                 |    |    |    |    |    |    |    |    |
original A:      | 00 | 11 | 22 | 33 | 44 | 55 | 66 | 77 |
                 |---------------------------------------|
                    0    1    2    3    4    5    6    7


                 _________________________________________
                 |    |    |    |    |    |    |    |    |
Ans: new A:      | 00 | 22 | 33 | 44 | 55 | 66 | 77 | 77 |
                 |---------------------------------------|
                   0    1    2    3    4    5    6    7
```

13. (5 points)

Complete the function `sorted()` that returns `true` if two `double` vectors are sorted in ascending order, and `false` otherwise. Declare any variables that you need. To get full credit, your function must have a single exit point.

```
bool sorted (vector<int> A, vector<int> B) {
   bool ordered = true;    // assume vectors are sorted

   for (int i = 0; i < A.size()-1 && ordered; i++)
     if (A[i] > A[i + 1]) ordered = false;
   for (int i = 0; i < B.size()-1 && ordered; i++)
     if (B[i] > B[i + 1]) ordered = false;
   return ordered;
}
```

14. (5 points)

Suppose you are doing a *bisection search* for the number `2000` in a 64-element vector that has been initialized to 0, 100, 200, ..., 6300:

```
vector<int> V (64);
for (int i=0; i<64; i++) V[i] = i*100;
```

Write down the low and high index for each stage of the bisection search until it finds the number `2000`. To give you a start, I have written down the first few:

```
 0   63
 0   30
16   30
```

Ans: [TCS 12.9]

```
16   22
20   22
20   20
```

15. (5 points)

Rewrite the following as a functionally equivalent `while` loop.

```
// M is an integer variable
for (i = 0; i < M; i++) {
     cout << i << " " << i*i << endl;
}
```

Answer:

```
i = 0;
while (i < M) {
     cout << i << " " << i*i << endl;
     i++;
}
```

16. (5 points)

Write an `int` function `firstNB (string S)` that returns the index of the first nonblank character in `S`. If `S` does not have any nonblank characters, then your function should return the length of `S`.

Ans:

```
int firstNB (string S) {
     for (int loc = 0; loc < S.size(); loc++)
          if (S[loc] != ' ') return loc;
     return loc;
}
```

There are many other ways to do this with either `for`-loops or `while`-loops.

17. (5 points)

Suppose the `Time` structure is declared:

```
struct Time {
    int hour, minute;
    double second;
    void print12() const;
};
```

Define the member function `print12()` to print the time in 12-hour format (with `AM` or `PM` appended as appropriate). For example, 07:15:00 should print as `7:15:00 AM` and 13:30:05 should print as `1:30:15 PM`. Note 00:45:00 should print as `12:45:00 AM`.

Ans:

```
Time::print12() const {
    int hr = hour;
    string AM_PM = " AM";
    if (hour >= 12) {
        hr -= 12;
        AM_PM = " PM";
    } else if (hour == 0) {
        hr = 12;
    }
    cout << hr << ':' << minute << ':' << second
        << AM_PM << endl;
}
```

18. (5 points)

Given the declaration of `Time` in question #17, define a constructor for `Time` that creates a time structure from input specified the 12-hour format. The constructor is declared:

```
Time (int h, int m, double s, string AM_PM);
```

The `AM_PM` parameter is either `"AM"` or `"PM"`. Write a definition (implementation) of this constructor.

Ans [TCS 11.7–11.8]:

```
Time::Time (int h, int m, double s, string AM_PM) {
    hour = h;
    minute = m;
    second = s;
    if (AM_PM == "PM") {
        hour += 12;
    } else if ((AM_PM == "AM") && (h == 12)) {
        hour = 0;
    }
}
```