Review Questions II — KEY

The following review questions are similar to the kinds of questions you will be expected to answer on Exam II (Nov. 23), which will cover LCR, chs. 1–7 and 10, and TCS, chs. 1–18. Make sure you do your best to answer each question before you look at the 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 16 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 `#includes`.

Ans: [LCS pp. 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)

What is the effect of the following statements?

```
int x, a = 11, b = 7;
x = a - b;
a = a - x;
b = a + x;
assert (b == 0);
```

Ans: It will generate an error message because the assertion "b == 0" is not true. [TCS 14.9]

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.

5. (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 };
```

6. (5 points)

Who can see member variables and functions that are declared:

(a) `public`
`Ans:` *Public members are visible to everyone .*

(b) `private`
`Ans:` *Private members are visible within the class in which they are declared, but nowhere*

else.

(c) `protected`
`Ans:` *Protected members are visible within the class in which they are declared and in its subclasses (derived classes), but nowhere else.*

7. (5 points)

In the space provided below, indicate what gets printed from the following program?

```
#include <iostream>
using namespace std;
void trace(int, int *);

int main()
{
  int x = 43, y = 15, *ptr = &x, *q = &y;

  trace(x, q);
  cout << "In main:  " << *ptr << "  " << *q << endl;
}

void trace(int x, int *q)
{
  int y;

  y = 2;

  *q += y;
  y = x / 10;
  cout << "In trace:  " << y << "  " << *q << endl;
}
```

Ans:

```
In trace:  4  17
In main:  43  17
```

8. (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

9. (5 points)

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

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

//Ans:

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

//end of ans.

```
  return v;
}
```

10. (5 points)

What does the word `virtual` mean in front of a function declaration?

Ans: It means that that function can be redefined in subclasses. [TCS 15.13]

11. (5 points)

In the Tic Tac Toe program discussed in ch. 10 of *Learning Computing with Robots* we discussed a function with the following declaration:

```
int lookahead (Board board, char player, bool MAX, int level);
```

Based on your understanding of the program, describe (1) the purpose of the function, (2) the meaning of each parameter, and (3) the meaning of the returned value.

Ans: [LCR pp. 276–8]

(1) The function computes a score for a board position based on looking ahead at possible moves.

(2) `board` *is the board being tested;* `player` *is X or O, the player whose position is being evaluated;* `MAX` *determines whether to maximize or minimize the evaluation; and* `level` *determines the amount of lookahead to do.*

(3) The function returns an evaluation of the board position.

12. (5 points)

Complete the following definition of a member function `sub()`, which subtracts two complex numbers. That is, `X.sub(Y)` should return the complex number obtained by subtracting the complex number `Y` from the complex number `X`. (Recall that the real part of the difference is the difference of the real parts, and the imaginary part of the difference is the difference of the imaginary parts.)

```
class complex {
public:
  double real, imag;
  complex sub (complex Y) {
    complex dif;
//Ans:
    dif.real = real - Y.real;
    dif.imag = imag – Y.imag;


//end of ans.
    return dif;
  } // end of sub
};
```

13. (5 points)

Write a definition to overload the "<<" operator to print out a complex number (as defined in question #12) . If X is the real part and Y is the imaginary part, it should print it in the form "`X + Yi`" if Y is nonnegative, and as "`X - Yi`" if Y is negative (that is, if X = 2 and Y = –1 it should print "`2 - 1i`". Treat the operator as a standalone function.

Ans: [TCS 15.8]

```
ostream& operator << (ostream& os, complex Z) {
  if (Z.imag >= 0) os << Z.real << " + " << Z.imag << "i";
  else os << Z.real << " - " << -Z.imag << "i";
  return os;
}
// It's OK to do endl on the outputs
```

14. (5 points)

Suppose that we extend the declaration of complex in question #12 to include prototypes for three constructors:

```
complex ();               // constructor a
complex (double);         // constructor b
complex (double,double); // constructor c
```

Implement these constructors as described below:

a. Implement `complex()` to create a complex number with 0 real and imaginary parts.
   Ans:
   `complex::complex() { real = imag = 0; } // OK if no complex::`

b. Implement `complex(X)` to create a complex number with real part `X` and imaginary part 0.
   Ans:
   `complex::complex(double X) { real = X; image = 0; }`

c. Implement `complex(X,Y)` to create a complex number with real part `X` and imaginary part `Y`.
   Ans:
   `complex::complex(double X, double Y) { real = X; imag = Y; }`

15. (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;`

16. (5 points)

Explain what is wrong with the following code segment? Look carefully!

```
int X = new int;
*X = 1;
*X = 0;
delete X;
```

Ans: It assigns a pointer to a non-pointer variable.

17. (5 points)

Given the following structure definitions, write an output command to print 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
```

18. (5 points)

Suppose that `Card` and `Deck` are defined as in question #17, 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.

```
}
```

19. (5 points)

What does *inheritance* refer to in the context of object-oriented programming?

Ans: It refers to the fact that a subclass (derived class) can automatically acquire member functions and member variables from the base class from which it is derived. [TCS 15.12]

20. (5 points)

Given the following definitions, write code to give Joe a 15% raise and his supervisor a 5% raise:

```
class employee {
public:
  string name;
  double salary;
  employee* supervisor;
};
employee* Joe;
/* omitted code to initialize the employee records,
   including Joe's */
// put your code here:
```

Ans: [TCS 16]

```
Joe->salary *= 1.15; // it's not necessary to use *=
Joe->supervisor->salary *= 1.05;
```

21. (5 points)

Define a subclass of `employee` (see question #20) called `manager` that has an additional public member variable, `subordinates`, which is a vector of pointers to `employee` records.

Ans: [TCS 15, 16]

```
class manager : public employee {
public:
  vector<employee*> subordinates;
}
```

22. (5 points)

State the *Information Hiding Principle*.

Ans: The implementer should have all the information needed to implement a module correctly, and nothing more. The user of a module should have all the information needed to use a module correctly, and nothing more. (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, Oct. 26]

23. (5 points)

Consider the following function `alike()`, which returns `true` if two integer vectors are alike, and `false` otherwise. The vectors are considered different if they have different lengths.

```
bool alike(const vector<int>& A, const vector<int>& B){
   bool same = true; // assume vectors are alike
   if (A.size() != B.size()) same = false;
   for(int i = 0; i < A.size() && same; i++)
     if(A[i] != B[i])  same = false;
   return same;
}
```

Modify this function so that it will work on vectors of any type (e.g., vectors of `int`s, vectors of `double`s, vectors of `string`s, and so on).

Ans: [TCS 17]

```
template <class TYPE>
bool alike(const vector<TYPE>& A, const vector<TYPE>& B){
   bool same = true; // assume vectors are alike
   if (A.size() != B.size()) same = false;
   for(int i = 0; i < A.size() && same; i++)
     if(A[i] != B[i])  same = false;
   return same;
}
```

24. (5 points)

Given the following declarations:

```
struct Node {
    int cargo;
    Node* next;
}
Node* data;
```

Write code to remove the second element of the linked list `data` and to return it to free storage.

(Assume that `data` points to a linked list with at least two `Node`s.)

Ans: [TCS 16.6, 18.8]

```
Node* second = data->next;  // pointer to 2nd Node
data->next = second->next;  // make 1st point to 3rd
delete second;              // delete 2nd Node
```

25. (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];
```

	0	1	2	3	4	5	6	7
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

26. (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
```

27. (5 points)

Given the following definition of `Complex`, define appropriate accessor functions to set and retrieve the private member variables. (Don't confuse this definition of `Complex` with the `complex` class used in other questions.)

```
class Complex {
private:
  double imag, real;
public:
  // you fill this in:
```

//Ans:

```
double getImag () { return imag; }
double getReal () { return real; }
void setImag (const double newImag) { imag = newImag; }
void setReal (const double newReal) { real = newReal; }
```

//end of ans. (const is optional)

```
} // end of Complex
```