CS 102 / ECE 206 — Spring '11

Review Questions for Final Exam — KEY

The following review questions are similar to the kinds of questions you will be expected to answer on the Final Exam, which will cover LCR, chs. 1–10 and TCS, chs. 1–18. 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 me by email or ask your Teaching Assistants in the Review Session. The actual exam will be about this length.

1. (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]

2. (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.*

3. (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]

4. (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: [TCS 16]

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

5. (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 (or parent) class from which it is derived. [TCS 15.12]

6. (5 points)

In the Tic Tac Toe program discussed in ch. 10 of *Learning Computing with Robots* there was 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.*

7. (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: [TCS 15.6]
```
    dif.real = real - Y.real;
    dif.imag = imag – Y.imag;
```
//end of ans.
```
    return dif;
  } // end of sub
};
```

8. (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. [TCS 16]

9. (5 points)

Write a definition to overload the “<<” operator to print out a complex number (as defined in question #7) . 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";
end; // It's OK to do endl on the outputs
```

Of course, there are various ways to write this.

10. (5 points)

Suppose that we extend the declaration of complex in question #7 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: [TCS 11.7]
   `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; }`

11. (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;
```

12. (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 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
```

13. (5 points)

Define a subclass of `employee` (see question #11) 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;
}
```

14. (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;
}
```

15. (5 points)

Given the following definition of `Complex`, define appropriate accessor functions to set and retrieve the private member variables. (Note that this is a different `Complex` class from question #7.)

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

//Ans: [TCS 14.2, 14.4]

```
     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
```

16. (5 points)

a. What are the two things that **new** does?

1.

Ans: allocates the requested storage from the help (freestore). [TCS 16.6]

2.

Ans: assigns the address of the space as a pointer.

b. What are the two things that **delete** does?

1.

Ans: returns the space to the heap (freestore) for re-use.

2.

Ans: the pointer is now undefined.

17. (5 points)

Examine the following code, and show what is displayed.

```
#include<iostream>
#include<vector>
using namespace std;

void copy_v (vector<int> v1, vector<int> v2) {
  v2 = v1;
}

main() {
  vector<int> v1(4);
  vector<int> v2(4);
  int i;

  for (i = 0; i < v1.size(); i++)  {
    v1[i] = i + 10;
    v2[i] = i + 20;
  }

  copy_v (v1, v2);

  for (int i = 0; i < v1.size(); i++)
    cout << v1[i] << "  " << v2[i] << endl;
}
```

Ans:

```
10  20
11  21
12  22
13  23
```

Note that `v2` *is* <u>*not*</u> *passed by reference. [TCS 8.6, 8.7]*

18. (5 points)

You have the following *class* definition.

```
class Example {
  public:
    Example();                     // default constructor
    Example(int, int, string);     // parameterized constructor

  private:
    string name;      // 20 actual characters maximum
    int x,            // range 1 through 30
        y;            // range 0 through 19
};
```

a. Write code for the default constructor where the integer member variables are set to 0, and the string member variable is set to the null string "".

Ans: [TCS 11.7]

```
         Example::Example(){
           x = y = 0;
           name = "";
 }
```

b. Write code for the parameterized constructor. Check for valid integers and appropriate string length. If out of range, set to the defaults.

Ans:

```
    Example::Example(int x_parm, int y_parm, string name_parm)
         {
           if (name_parm.size() <= 20) name = name_parm;
           else name = "";

           if (x_parm >= 1 && x_parm <= 30)  x = x_parm;
           else x = 0;

           if (y_parm >= 0 && y_parm <= 19)  y = y_parm;
           else y = 0;
```

```
          }
```

19. (5 points)

Complete function `counter()` to find the largest integer in a vector `A` (of size greater than zero) that is sorted in ascending order. Return both the value of the largest element and the number of times it occurs.

```
          void counter (int* max, int& count, A)
```

Ans: [TCS 8.7, 16.4]

```
     void counter(int* max, int& count, vector<int> A) {
       const int N = A.size(); // const is optional
       *max = A[N-1];
       count = 1;

       for (int i = 0; i < N - 1; i++)
         if (A[i] == *max)  count++;
     }
```

There are more efficient solutions (e.g., start with `A[N-1]` *and work backward until the beginning or an element not equal to* `max` *is found).*

20. (5 points)

A *palindrome* can be defined as a string that reads the same backward or forward. For example, `"abcba"` and `"1234 4321"` are palindromes. Complete the function `pal()` that takes a string as input and returns the length of the string if it is a palindrome and zero otherwise. Declare variables that you need.

Ans:

```
     int pal (string str) {
         int alike;
         const int length = str.size();
         alike = length;              // assume chars are alike
         for(int i=0; i < length/2 && alike; i++)
             if(str[i] != str[length - i - 1])  alike = 0;
         return alike;
     }
```

Of course there are other correct solutions, but make sure they work for both odd- and even-length strings, and if the string is short (0, 1, or 2 characters).