CPSC 101 Winter 2007
Review Questions—02 April 2007

Shorter Answers

Blanks may stand for one or more words.

1. Some physical subsystems of a typical computer include __________, the CPU, __________, __________, and input/output devices. The connections between these components are called __________.

2. The four major regions of memory from the point of view of a C++ program are __________, __________, __________, and __________.

3. The phases of a compilation of a single-file C++ program to produce an executable are __________, compilation, and __________. The compilation phase produces ____________ files.

4. The three special kinds of member functions that don't have return types are ____________, destructors, and _______________.

5. The four (or five) phases of an object's lifetime are: ____________, construction, (do something useful), ____________, and ____________.

Short Answers

6. Does a C++ memory diagram typically capture what is going on in the entire memory of a computer? Why or why not?

7. (a) What objects are allocated before `main` begins?
   (b) What objects are de-allocated after `main` ends?
   (c) What objects are destroyed at a time explicitly chosen by the programmer ?

8. In what order are the member variables of an object constructed? Does this depend on the colon list? What happens to variables not mentioned in a colon list?

9. Explain why a stack is necessary.

10. What kinds of information are stored on the stack?

11. What is the difference between the memory diagram of the stack frame of a static member function and that of non-static member function?

12. *Heisenbugs* are indicative of what kind of programming error?

13. Suppose that you isolate an error that is causing your program to crash, and that it is occurring when you `delete` an object on the heap. Further investigation shows that this is a perfectly legitimate object to delete. Explain what is likely happening.

14. When are zero-argument constructors compiler-supplied?

15. What other constructors are sometimes compiler supplied? When?

16. Name all of the kinds of constructors that the compiler treats specially and the circumstances under which it does so.

17. Explain the `explicit` keyword.

18. What is special about the signature of a copy constructor?

19. What is a *compilation unit*?

20. How does inheritance affect memory diagrams?

21. Declare an array of four pointers to functions that take a `string` argument and return `void`.

22. (a) Define what we mean by the *apparent* and *actual* type of an object.

(b) How does this relate to `virtual` functions?

23. Suppose that we have:

```
#include <iostream>
class A           { public:         char f(void) const {return 'a' ;} } ;
class B : public A { public: virtual char f(void) const {return 'b' ;} } ;
class C : public B { public: virtual char f(void) const {return 'c' ;} } ;

int g(const A& a) {return a.f() ;}
int h(const B& b) {return b.f() ;}

int main ()
{
    using std::cout ; using std::endl ;
    C cc ; B bb ;
    cout << g(cc) << g(bb) << endl ;
    cout  << h(cc) << h(bb) << endl ;
    return 0 ;
}
```

What does this program print? Why?

24. Explain how the `sizeof` an object interacts with pointer arithmetic.

25. Complete the following table:

SYMBOL	In a declaration	In an expression
[2]	*an array of size 2*	*the third element of*
&		
*		

26. Show how to use the `new` operator to:

(a) create a new `Date` object on the heap using the default constructor;

(b) create a new `Date` object on the heap using constructor arguments `(2007,04,20)`;

(c) create a new array of 13 `Date` object on the heap.

(d) Which constructor is used in the latter case?

27. Show how to declare the class `PartTime` so that it is dervied from both `Employee` and `Casual`.

28. In terms of inheritance and assignment and copy construction, what is *slicing*?

29. Show how to make all of the member functions of the class `GoodNeighbor` to be friends of the class `MisterRogers`.

True and False

Circle **TRUE** or **FALSE** as appropriate. Questions that don't clearly indicate *one* choice shall be marked wrong.

1. Friendship is granted, not taken. **TRUE** **FALSE**
2. Friendship can be inherited from a base class. **TRUE** **FALSE**
3. Friendship is symmetric. **TRUE** **FALSE**
4. Friendship is reflexive. **TRUE** **FALSE**
5. Friendship is transitive. **TRUE** **FALSE**
6. There is an automatic conversion from an ℓ-value to an r-value. **TRUE** **FALSE**
7. C++ supports multiple inheritance. **TRUE** **FALSE**

8. Most object-oriented programming languages do not have virtual functions. **TRUE** **FALSE**

9. Most object-oriented programming languages do not have the `virtual` keyword. **TRUE** **FALSE**

10. There is an automatic conversion from an r-value to an ℓ-value. **TRUE** **FALSE**

11. The apparent type of an object can be determined at compile-time. **TRUE** **FALSE**

12. The actual type of an object can be determined at compile-time. **TRUE** **FALSE**

Memory Diagrams