CSc 372, Spring 1996
Final Examination Solution Key

Problem 1 (10 points):

What is the difference between a class and an object?

> An object is an instance of a class. In software, classes are used to create objects, specifying structure and behavior. In the real-world, classes are used to group objects.

What is the proper way to view the relationship between function members and data members in a C++ class?

> Data members exist solely to support the operation of the function members. A class with no function members should have no data members.

Under what circumstances should a data member in a C++ class be public?

> Never.

What is the difference between the class relationships of inheritance and containment?

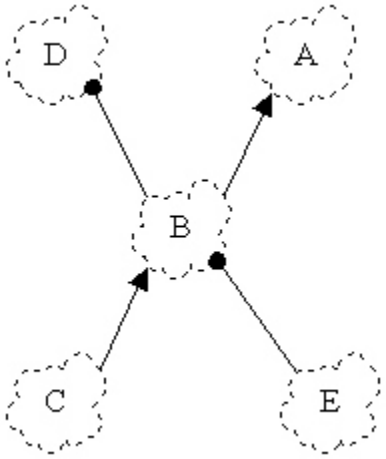
> Containment is the "has-a" relationship; it should be used when instances of a class in some sense contain instances of another class. The containment might be physical or logical. Inheritance is the "is-a" relationship; it should be used when an instance of a class can be thought of as being an instance of another class.

As described by the instructor, what is the primary benefit of inheritance?

> The ability to write code in terms of something general, such as a geometric shape, and then use that code with instances of derived classes that are specialized, such as circles and triangles.

Problem 2 (5 points):

Write a set of C++ class declarations that capture the relationships shown in this Booch Notation class diagram:




```
class A {};

class E {};

class B: public A {
     E itsE;
     };

class C: public B {};

class D {
     B itsB;
     };
```

Problem 3 (20 points):

```
#include <iostream.h>
#include "string.h"

class ReplStr {
  public:
     ReplStr() {}
     ReplStr(String s, int n) {
          itsStr = s.Repl(n);
          }
     int Length() { return itsStr.Length(); }

     String GetString() { return itsStr; }

     int operator==(ReplStr rhs) {
          return itsStr == rhs.itsStr;
          }

     int operator!=(ReplStr rhs) {
          return !(this->GetString() == rhs.itsStr);
          }

     ReplStr operator+(ReplStr rhs)
     {
          String r = GetString() + rhs.GetString();

          return ReplStr(r, 1);
     }
```

```
  private:
     String itsStr;
     };

ostream& operator<<(ostream& o, ReplStr s)
{
     o << s.GetString();
     return o;
}
```

(5) Describe a pattern of usage for which your implementation would have poor performance characteristics and explain the difficulty. If you believe your implementation has uniformly good performance characteristics, make an argument to support that claim.

> The choice to represent a `ReplStr` as a `String` makes for a simple implementation but could be very inefficient. If nothing else, construction of the `String` could be deferred until the first time `GetString` is called. (Note that length can be calculated given only the base and replication count.)

Problem 4 (5 points):

Fully implement classes `Bicycle`, `Tricycle`, and the function `CountWheels`.

```
class Cycle {
  public:
    Cycle(char *owner) {
        itsOwner = new char[strlen(owner)+1];
        strcpy(itsOwner, owner);
                }

        virtual int GetNumWheels() = 0;
        char *GetOwner() { return itsOwner; }
  private:
        char *itsOwner;
        };

class Bicycle: public Cycle {
  public:
        Bicycle(char *owner) : Cycle(owner) {}
        virtual int GetNumWheels() { return 2; }
        };

class Tricycle: public Cycle {
  public:
        Tricycle(char *owner) : Cycle(owner) {}
        virtual int GetNumWheels() { return 3; }
        };

int CountWheels(Cycle *cycles[])
```

```
{
        int wheels = 0;

        for (Cycle **cp = cycles; *cp; cp++)
                wheels += (*cp)->GetNumWheels();

        return wheels;
}
```

Problem 5 (5 points):

Write a Prolog predicate `flip(L1,L2)` that if given a list such as `[a,b,c,d,e,f]` as `L1` it will instantiate `L2` to be the list `[b,a,d,c,f,e]`. That is, it flips the position of each element in a list on a pair-wise basis. `flip` should fail if given a list with an odd length.

```
flip([],[]).

flip([X1,X2|XS],[X2,X1|NewXS]) :- flip(XS, NewXS).
```

Problem 6 (5 points):

State in your own words the relationship expressed by each of the rules of the `append` predicate:

```
append([],L,L).
```

The list `L` appended to the empty list is the list `L`.

```
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

The list `L2` appended to the list having head `X` and tail `L1` is the list with head `X` and tail `L3` where `L3` is the result of appending `L2` to `L1`.

Problem 7 (5 points):

Write a Prolog predicate `trim(L, SL, NL)` that describes the relation that the list `NL` is the list `L` with the list `SL` removed if `SL` is a prefix or suffix of `L`. Note that if `SL` is a both a prefix and a suffix, two alternatives should be generated.

```
trim(L, SL, NL) :- append(SL, NL, L).
trim(L, SL, NL) :- append(NL, SL, L).
```

Problem 8 (5 points):

Do EITHER part (8a) or part (8b), but not both. If you work on both, CLEARLY mark which solution you wish to have graded.

(8a) Write a predicate `maxint/2` that if given a list of integers will find the largest element in the list. `maxint` should fail if given an empty list.

```
maxint([X],X).

maxint([X1,X2|XS], M) :-
    X1 > X2, maxint([X1|XS], M).

maxint([X1,X2|XS], M) :- X1 =< X2, maxint([X2|XS], M).
```

(8b) Write a predicate `sum_check/2` that determines if a given integer is the sum of a list of integers.

```
sum_check(0,[]).
sum_check(Sum, [H|T]) :-
     sum_check(TSum, T), Sum is H + TSum.
```

Problem 9 (5 points):

Write a Prolog predicate `find_missing/3` that can be called with any two arguments instantiated and if the two supplied arguments are equal, the uninstantiated argument is instantiated to the value of the other two.

```
three_eq(X,X,X).
```

Problem 10 (5 points):

(1) Write a Prolog query that expresses *Is there is a person who knows both Bob and Mary?*:

```
knows(X,bob), knows(X,mary).
```

(2) Write a Prolog query that expresses *Does Mary know anyone who works on a different shift than she?*:

```
knows(mary,X), shift(mary,S), \+shift(X, S).
```

(3) Define a new clause for `knows/2` that expresses *A person knows everyone who works on the same shift.*

```
knows(X,Y) :- shift(X,S), shift(Y,S), X \== Y.
```

(4) Define a predicate `by_shift` that prints a list of employees by shift.

```
by_shift :- report, fail.

report :- write('Day:'), nl, shift(day), nl.
report :- write('Night: '), nl, shift(night), nl.

shift(X) :- shift(P,X), write('   '), write(P).
```

Problem 11 (3 points):

Write an ML function `allsame(L)` of type `''a list -> bool` that returns true if all the elements in a list are equal and false otherwise. `allsame([])` should return false.

```
fun allsame([]) = false
 |  allsame([x]) = true
 |  allsame(x1::x2::xs) = x1 = x2 andalso allsame(x2::xs)
```

Problem 12 (2 points):

Consider this function definition:

```
fun f(x,g) = x::g(x-1)
```

What types would ML infer for:

```
f?   int * (int -> int list) -> int list

g?   int -> int list

x?   int
```

Problem 13 (5 points):

Write an Icon program `tac` to read a text file on standard input and print on standard output the lines of the file in reverse order—last line first; first line last.

```
procedure main()
    lines := []

    while push(lines, read())

    while write(get(lines))
end
```

Problem 14 (20 points):

If you don't know this by now, I doubt that telling you one more time will help...

Extra Credit Problems

(1 point) Andrew Koenig's paper, *An Anecdote About ML Type Inference*, describes an incident in which ML's type inferencing system produced a surprising result. What was that result?

It was detected that a program would not terminate.

(2 points) Name five programming languages that originated before 1980.

ML, Icon, C, C++, and Prolog

(2 points) In C, write a recursive version of the library function `strlen(char*)`. You may not use any library functions, or perform any assignment, augmented assignment, or increment/decrement operations. In other words, write `strlen` using a functional style.

```
int strlen(char *p)
{
     if (!*p)
          return 0;
     else
          return 1 + strlen(p + 1);
}
```

(1 point) Who was the person that first described the notion of partially evaluated functions, such as that provided with currying in ML?

Schonfinkel. (Curry came along later.)

(1 point) Name a popular operating system in which Prolog is utilized.

Windows NT

(5 points) Do a great job on question 14.

Did you?