CSc 372, Fall 1996
Mid-Term Examination Solutions

Problem 1: (2 points each; 4 points total)

State the type of each of the two following expressions, or if the expression is not valid, state why.

```
(1,2,(1=2,1+2),"x")
```

```
int * int * (bool * int) * string
```

```
[[],[1],[1,2]]
```

```
int list list
```

Problem 2: (2 points each; 8 points total)

Consider this ML function definition:

```
fun f(a,b,c) = (b, a(b), c(a)) = ("x", 2, [1])
```

State the type that will be deduced for each of the following: (2 points each)

a: `string -> int`
b: `string`
c: `(string -> int) -> int list`
The type of the result produced by `f`: `bool`

Problem 3: (4 points)

Write a function `f` that has the type `int -> int -> int -> int -> bool`. You may use literals (constants) but you may not use any explicit type specifications such as `x:int`.

```
fun f a b c d = a + b + c + d = 1 (* A. Freshwater *)
```

- or -

```
fun f 1 2 3 4 = true
```

Problem 4: (12 points)

Write a function `ints_to_strs(L,c)` that for the `int list L` and the `string c` produces a list of strings where the *i*th element in the resulting list is a string composed of *N* replications of `c` where *N* is the *i*th element in the input list.

Most solutions took this form:

```
fun repl(s, 0) = ""
 |  repl(s, n) = s ^ repl(s, n-1);

fun ints_to_strs([], _) = []
 |  ints_to_strs(x::xs, s) = repl(s, x) :: ints_to_strs(xs, s)
```

Problem 5: (12 points)

Write a function `len(L)` of type `string list list -> int` that produces the total number of characters in all the strings in the lists contained in `L`. (12 points)

```
fun len(L) = size(reduce(op^,reduce(op@, L))) (* P. Holland *)
```

Problem 6: (10 points)

Write a function `pair(L)` that accepts an `'a list` as an argument and produces a list of tuples containing consecutive pairs of elements from the list `L`.

```
exception OddLength;

fun pair([]) = []
 |  pair([x]) = raise OddLength
 |  pair(x1::x2::xs) = (x1,x2)::pair(xs)
```

Problems 7 and 8: (26 and 10 points, respectively)

```
class Food {
  public:
    virtual int GetUnits() = 0;
    };

class Burger: public Food {
  public:
    int GetUnits() { return 25; }
    };

class Fries: public Food {
  public:
    int GetUnits() { return 15; }
    };
```

```
class Eater {
  public:
    Eater(int capacity)
     : itsMax(capacity), itsBurps(0), itsValue(0) {}

    void Eat(int units) {
        itsValue += units;
        while (itsValue >= itsMax) {
            cout << "burp!" << endl;
            itsBurps++;
            itsValue -= itsMax;
            }
        }

    void Eat(Food *fp) {
        Eat(fp->GetUnits());
        }

    int BurpCount() const { return itsBurps; }

  private:
    int itsMax, itsBurps, itsValue;
    };

ostream& operator<<(ostream& o, Eater& E)
{
    o << "Eater at " << &E << " has burped " << E.BurpCount()
           << " times" << endl;

    return o;
}
```

Problem 9: (2 points each; 14 points total)

Answer each of the following questions related to object-oriented programming in general and C++ in particular.

What is the difference between a class and an object?

An object is an instance of a class.

What is the purpose of a constructor?

To initialize an object.

What is the purpose of a destructor?

To reclaim resources acquired during the lifetime of the object.

If a class has no member functions, how many data members should it have?

"Zero" is a reasonable answer, but with proper rationale the answer "any number" is also acceptable.

Challenge or defend this statement: In C++, one possible reason to have a public data member is to avoid the overhead of calling a function to access that data.

> Using an inline member function maintains encapsulation but without any performance penalty.

Challenge or defend this statement: The purpose of member functions is to provide well-controlled access to data members.

> No, the purpose of member functions is to provide the required behavior for the class. Data members merely support member functions.

Describe a benefit provided by using inheritance.

> Being able to treat an object as being something more general than it really is.

Optional Extra Credit Problems:

What is a practical reason to prefer pattern matching rather than using the `hd` and `tl` functions in SML? (2 points)

> With pattern matching a non-exhaustive match can indicate a situation where a usage of `hd` could produce an exception. Pattern matching can also be argued for on the basis of efficiency and clarity.

Write the minimum amount of code necessary to compile and run this code: (4 points)

> The intended response was "`class X {};`" but a number of persons observed that "`typedef char X;`" or "`#define X char`" was enough.

Could C++ be used effectively for functional programming? (5 points)

> This could be argued successfully either way.

Write in C a version of `int strcmp(char *, char *)` that uses no assignment operators of any form, nor the `++` or `--` operators. Recall that `strcmp` returns 0 if the strings are equal and a non-zero value if the strings are not equal. (3 points)

```
int strcmp(char *a, char *b)
{
    if (*a == '\0' && *b == '\0') return 0;
    else if (*a == '\0' || *b == '\0') return 1;
    else if (*a == *b) return strcmp(a+1, b+1);
    else return 1;
}
```