CSc 372, Spring 1997
Mid-term Examination Solutions

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

State the ***type*** of each of the following expressions, or if the expression is not valid, state why. For example, the type of the expression `3+4` is `int`.

```
([1, 2], [3.0, 4.0])

     int list * real list


(map size) (explode "abcd")

     int list


("x", (2, "y"), ([3, "z"]))
```

Invalid: `[3, "z"]` attempts to form a heterogeneous list

```
(length, [size, length o explode])

     ('a list -> int) * (string -> int) list
```

Problem 2: (2 points)

Consider this ML function definition: `fun f(x) = [x 1, 2]` What type will be deduced for `f`?

```
(int -> int) -> int list
```

Problem 3: (1 point each; 5 points total)

Consider these two ML function definitions:

```
fun f(_) = 1
fun g(a,b,c) =  if a = f(c) then b else [c,a]
```

State the type that will be deduced for each of the following:

a: `int`
b: `int list`
c: `int`
The return type of `f`: `int`
The return type of `g`: `int list`

Problem 4: (3 points)

Write a function `f` that has the type `int -> int list -> bool`.

```
fun f 1 [1] = true
```

Problem 5: (9 points)

Write a function `tossbig(L,N)` of type `string list * int -> string list` that produces a list consisting of the elements in `L` that have a size less than `N`.

```
fun tossbig([], _) = []
 |  tossbig(s::ss, n) =
        if size(s) < n then s::tossbig(ss, n)
                       else tossbig(ss,n)
```

Or, a non-recursive solution using `filter` from the slides:

```
fun tossbig([], _) = []
 |  tossbig(L, n) = filter(fn(s) => size(s) < n, L)
```

Problem 6: (9 points)

Write a function `samesums` of type `(int * int) list -> bool` that produces true if adding the two elements of each tuple in turn produce the same sum.

```
fun samesums [] = true
 |  samesums [_]  = true
 |  samesums ((a,b)::(c,d:int)::ts) =
      a+b = c+d andalso samesums((c,d)::ts)
```

Problem 7: (12 points)

Write a function `block(w,h)` of type `int * int -> unit` that prints a block of `x`'s having a width of `w` and height of `h`. You may assume that `w` and `h` are greater than or equal to 1. **Your solution must be NON-RECURSIVE in nature.**

When this problem was written, an answer like this was envisioned:

```
fun block(w,h) =
    let
        fun row _ =
           implode (map (fn(x) => "x") (m_to_n(1,w))) ^ "\n"
    in
        print(implode ((map row) (m_to_n(1,h))))
    end
```

However, some persons observed that the solution could be simplified with `repl`, from the second assignment:

```
fun block(w, h) = print(repl(repl("x", w) ^ "\n", h))
```

If I had considered that `repl` might be used, I would have specifically excluded it, but I since I did not, I considered solutions using `repl` to be perfectly acceptable, even though that essentially side-stepped the purpose of the problem.

Problem 8: (6 points)

Write a function `c_block` of type `int -> int -> unit` that behaves like `block`, but that allows partial application. You may use `block` in your solution.

```
fun c_block w h = block(w,h)
```

Problem 9: (30 points)

In this problem you are to implement a C++ class that models a television set that has a list of favorite channels, each with a preferred volume. ...

```
class TV {
    ...
  private:
    int itsFavs[84];
    int itsCurChan;
    int itsCurVol;
    };

TV::TV()
  : itsCurChan(2), itsCurVol(5)
{
    for (int i = 2; i < 84; i++)
        itsFavs[i] = -1;
}

int ckRange(int low, int high, int val)
{
    if (val < low || val > high)
        return 0;
    else
        return 1;
}

void TV::SetVol(int vol)
{
    if (!ckRange(0, 10, vol))
        return;

    itsCurVol = vol;
}

void TV::SetChan(int chan)
{
    if (!ckRange(2, 83, chan))
        return;

    itsCurChan = chan;
}

void TV::SetFav()
{
    itsFavs[itsCurChan] = itsCurVol;
}

void TV::SetFav(int vol)
{
    if (!ckRange(0, 10, vol))
        return;

    itsFavs[itsCurChan] = vol;
}

void TV::NextFav()
{
    int i;
    for (i = itsCurChan+1; i <= 83; i++)
        if (itsFavs[i] != -1) {
            SetVol(itsFavs[i]);
            SetChan(i);
            return;
            }

    for (i = 2; i < itsCurChan; i++)
        if (itsFavs[i] != -1) {
            SetVol(itsFavs[i]);
            SetChan(i);
            return;
            }
}

void TV::Status()
{
    cout << "Channel is " << itsCurChan << ", volume is "
        << itsCurVol << endl;
}
```

Problem 10: (5 points)

With our `String` class in mind, overload the `%` operator so that an expression such as `s % c` produces an `int` that is the zero-based position of the first occurrence of the character `c` in the `String` `s`.

```
    int operator%(const String& s, char c)
    {
        char *p = strchr(s.GetPtr(), c);

        if (p == 0)
            return -1;
        else
            return p - s.GetPtr();
    }
```

Problem 11: (2 points each; 6 points total)

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

(a) What is the difference between a class and an object?

An object is an instance of a class.

(b) As you know, the destructor for a class `X` is a method named `~X`. The rationale for this choice of name is that "destruction is the complement of construction". However, the instructor has contended on several occasions that the relationship between constructors and destructors is in fact somewhat asymmetrical. Describe that asymmetry.

The responsibility of a constructor is to initialize an object. The destructor's responsibility is to surrender resources acquired during the lifetime of the object.

(c) What is the key benefit provided by in-line functions in C++?

In-line functions provide access that is just as fast as directly referencing data members, but without loss of encapsulation.

Problem 12: (9 points)

In this problem you are to implement a class called `Ring` that is a subclass of the `Shape` class described in the slides.

```
class Ring: public Shape {
    public:
        Ring(double inner, double outer)
        : itsInner(inner), itsOuter(outer) {}

        double Area() {
            return itsOuter.Area() - itsInner.Area();
            }

        double Perimeter() {
            return itsOuter.Perimeter();
            }

        void Print(ostream& o) {
            o << "Ring; area = " << Area() << ", perim = "
                << Perimeter() << endl;
            }

    private:
        Circle itsInner, itsOuter;
        };
```

No changes are required in `SumOfAreas` or `Biggest`.