Matthew Kelly
Dr. George Rudolph
CSIS618
18 February 2009
Homework 5

Question 5.1

OCaml supports integers (4 byte), floating point numbers (4 byte), Booleans (4 bytes), characters and character strings for its basic data structures.

Question 5.2

Booleans appear to be ordered in OCaml where true is greater than false.
The boolean type in OCaml is not directly convertible to an interger. As the language is mostly functional and the boolean data type is rarely explicitly used (rather, function evaluations are analyzed), I do not believe that having OCaml's boolean type be an integer would be particularly useful. Through coercions, a bool can be converted to a string then the string to an int, but I believe that the variable loses its boolean meaning in doing this, so the string to int conversion would not result in "1" like we would expect from a bool being converted to an int.

Question 5.5

I wrote the following script to determine this. The decimal places are included for legibility.
0.2 : 0.00110011001100110011001100110011001
0.5 : 0.10000000000000000000000000000000
0.3 : 0.01001100110011001100110011001100110
1 : 1.00000000000000000000000000000000

```
ns = [.2,.5,.3,1.0];
str = "";
for(ndex = 0; ndex<ns.length; ndex++){
	n=ns[ndex];/*The value for which we're solving */
	str+= n+" : ";
	k=n; /* k = amount of n remaining, used to preserve n */
	y=1; /* exponent will be inverse of this, e.g. first 1/2 is first evaluated */

	for(i=0; i<32; i++){
		if((1/y)<=k){
			str+="1";
			k=k-(1/y);
		}
		else {str+="0";}

		y=y*2; /*divide 1/y by 2, e.g. from 1/2 to 1/4 to 1/8, etc */
		if(i==0){str+=".";}
	}
```

```
        str+="<br />";

}
document.write(str);
```

Question 5.9

Big-endian and little-endian are different orderings used to represent data. Big-endian architectures store data from the most significant bit to the least significant and little endian architectures do the opposite. x86 is an example of a little-endian architecture. Motorola's 68k processors used big-endian. The values in table 5.2 are interpreted differently depending on the architecture used. If the data is interpreted from the perspective of a big-endian architecture, for example, the first four bits of the float example, 0100, are the most significant; that is, they represent the 2^{31}, 2^{30}, 2^{29} and 2^{28} bits. A little endian perspective would have these bits be the least significant (assuming linear processing) of 2^0, 2^1, 2^2 and 2^3.

Question 5.16

Dynamic arrays are arrays that can be resized to support a variable number of elements. OCaml does not support dynamic arrays but does support appending items onto an array. In doing so, the space used to hold the original array is (likely) not used for the new array but instead the elements of the resulting append operation are stored in a new location.

Question 5.17

Associative arrays allow an array to be defined with potentially non-numerical keys, e.g. in a scientific name array, looking up the value in the array at position "Acer Rubrum" might yield "Red Maple". OCaml natively supports associative arrays. An example usage is:

```
let scientificNames = [

        "Acer Rubrum","Red Maple";
        "Quercus virginiana","Southern Live Oak";
        "Casuarina equisetifolia","Northern Australian Pine"];;
```

Recalling a value by hash is as easy as:

```
List.assoc "Acer Rubrum" scientificNames;;
```