University of Nevada, Reno

CS 326 — Programming Languages

Assignment #5

Joshua Gleason

Instructor:
Dr. Mircea Nicolescu

November 2, 2010

1. The reason that Boolean types are represented on a byte in memory is because in memory, bits do not have their own addresses. The smallest memory location that is addressed is a byte. This means that although only a bit is needed in theory, in practice this is not realized due to this hardware limitation.

2.

```
#include <stdio.h>

typedef union
{
  int ival;
  float fval;
} foo;

void main()
{
  foo obj;

  obj.ival = 1065353216;  // 1.0
  printf("%f\n",obj.fval);
  obj.ival = 1073741824;  // 2.0
  printf("%f\n",obj.fval);
}
```

Output:

```
1.000000
2.000000
```

This demonstrates changing the bits in `foo` to represent the `int` values 1065353216 then 1073741824 respectively. The values are then printed as if the bits represented a `float` rather than an `int`. The bit value for 1065353216 as a 32-bit `int` is

`0011 1111 1000 0000 0000 0000 0000 0000`

which in floating point is $2^{01111111_2 - 127_{10}} = 2^0 = 1$

The second value is

`0100 0000 0000 0000 0000 0000 0000 0000`

which in floating point is $2^{1000000_2 - 127_{10}} = 2^1 = 2$

(Note: Assumes 32-bit `float` and `int` types)

3. (a) `a,b,c` and `d` are all structural equivalence to each other.
   (b) `a` and `b` are strict name equivalence to each other, `c` and `d` are not.
   (c) `a,b` and `c` are loose name equivalences to each other, `d` is not.

4. Although memory is allocated for an instance of `Foo` in the `allocate_node` function, the parameter is passed by value and therefore, the address of the allocated memory is lost when the function ends. In the main function `p` still contains the garbage value that it was initialized to by default. The program is attempting to de-reference a garbage memory location which is what causes the run-time error.

```
typedef struct
{
  int x;
  int y;
} Foo;

void allocate_node (Foo ** f)
{
  (*f) = (Foo *) malloc ( sizeof(Foo) );
}

void main()
{
  Foo * p;
  allocate_node (&p);
  p->x = 2;
  p->y = 3;
  free(p);
}
```

By passing an address of a pointer (double pointer) the `Foo` pointer value can be modified. In `main` the address of `p` is passed using the `address of (&)` operator.