Hayden Shelton
9/8/14

CS4400
Problem Set 2

2.7.1

a. The problem with this code is that it ignores all digits to the left of the byte in question. In the case of negative numbers, those digits are necessary to represent the number.

b.

```
45
46
47  typedef unsigned packed_t; //packed two's complement number
48
49
50  int xbyte (packed_t word, int bytenum)
51  {
52      //println(int2bin(word));
53      //get # of bits to shift over so that word[bytenum] is in slot 0
54      int pos = bytenum <<3;
55      //shift over by pos
56      int relaventBytes = (word >> pos);
57
58      //if the word is negative we need to do more work
59      //shift over all the way to the right
60      packed_t shifted = word >> 31;
61      int sign = shifted & 1;
62      println(int2bin(sign));
63
64      if (sign == 1)
65          return (~relaventBytes) +1;
66
67      else
68      {
69          return relaventBytes;
70      }
71
72  }
73
```

2.7.4

```
75  int tsub_ok (int x, int y)
76  {
77      //if both integers have contents in the last bit we could have problems
78      int first = (x >>31) & 80000000;
79      int second = (y >>31) & 80000000;
80
81      return !(first && second);
82
83
84
85  }
```

2.7.6

```
 93
 94  void problem3 ()
 95
 96  {
 97      //k = 17
 98      int result = x <<4;
 99      result ++x;
100
101      //k = -7
102      int result = x - (x<<1);
103      result = result <<3;
104      result += x;
105
106
107      // k = 60
108      int result = x <<6;
109      int temp = x <<2;
110      result -= temp;
111
112      //k = -112
113      int result = x - (x<<1);
114      result = result <<7;
115      result += x <<4;
116
117
118  }
119
```

2.7.7

```
 86
 87  int divide_power2(int x, int k)
 88  {
 89      int t = (1 <<k)>>1;
 90
 91      return (x-t) >>k;
 92
 93  }
 94
 95
 96
```