High Integrity Software Engineering, Workshop 9 solutions

From the Parr & Pelzl textbook:

1. Exercises 1.7 & 1.8 (modular arithmetic).
    a. Ans: The tricky one is 1.7(4) – the elements without multiplicative inverses are elements that are not coprime with the modulus, in particular 2 (mod 4), and 2, 3 and 4 (mod 6), and also 0 mod anything. Take 2 (mod 6) for example. This can never have a multiplicative inverse, which would be a number n s.t. 2 * n =1 (mod 6). But 2*n is always a multiple of 2, which always leaves a remainder that's a multiple of 2 when divided by 6, so it's not possible to make 2 * n equal 1. Similar argument for anything that shares a factor with the modulus. None of them for 5 because it's prime.
    b.
2. Read 1.10 for the definition of Euler's phi function. Then compute ϕ(1319 * 1321). Hint: a little thinking will save you a lot of computing.
    a. Ans: If p is prime, then ϕ(p) = p-1. If p and q are prime, then ϕ(pq) = (p-1)(q-1), because there are pq elements, minus p multiples of q, minus q multiples of p, plus an extra 1 because we subtracted pq itself twice.
3. Exercise 7.3, 7.9
    a. Ans: I hope 7.3 is straightforward.
    b. 7.9: Alice gets Bob's public key, K_B. She generates a summetric key k, then she sends Bob k RSA-encrypted with K_B. (There are other correct solutions to this one, but I think this is the simplest.)
4. Exercise 10.5 & 10.6.
    a. Ans: Hope these are easy; just use plain RSA.
5. Forging RSA signatures.
    a. See Boneh & Shoup 13.3.1 "The importance of hashing", p.523-4.
6. Forging MACs.
    a. See Boneh & Shoup p.303, "Append the key."

```
function (doc) {
  var category = ["Education", "Shopping"];
  var suburb_melbourne = ["Melbourne", "Carlton"];
  var suburb_sydney = ["Sydney"];
  var result = {neutral_Melb: 0, positive_Melb: 0, negative_Melb:0, neutral_Syd: 0,
positive_Syd: 0, negative_Syd: 0 };
  var location;
  for (k=0; k< doc.doc.length; k++) {
    if (suburb_melbourne.indexOf(doc.doc[k].location.suburb) !=-1) {
      location = "Melb";
      for (i=0; i < category.length; i++) {
        for (j=0; j< doc.doc.length; j++) {
```

```
        if (doc.doc[j].lifestyle.indexOf(category[i]) !=-1) {
          switch (doc.doc[j].sentiment) {
            case "neutral": result.neutral_Melb++; break;
            case "positive": result.positive_Melb++; break;
            case "negative": result.negative_Melb++; break;
          }
        }
      }
      emit(location, {"liftstyle": category[i], "neutral": result['neutral_' + location], "positive":
result['positive_' + location], "negative": result['negative_' + location]});
    }
   }
   if (suburb_sydney.indexOf(doc.doc[k].location.suburb) !=-1) {
    location = "Syd";
    for (i=0; i < category.length; i++) {
     for (j=0; j< doc.doc.length; j++) {
      if (doc.doc[j].lifestyle.indexOf(category[i]) !=-1) {
        switch (doc.doc[j].sentiment) {
          case "neutral": result.neutral_Syd++; break;
          case "positive": result.positive_Syd++; break;
          case "negative": result.negative_Syd++; break;
        }
      }
     }
     emit(location, {"liftstyle": category[i], "neutral": result['neutral_' + location], "positive":
result['positive_' + location], "negative": result['negative_' + location]});
    }
   }
  }
}


function (keys, values, rereduce) {
  var result = { Education: {neutral: 0, positive: 0, negative:0}, Shopping: {neutral: 0, positive:
0, negative:0} }
  var result_syd = { Education: {neutral: 0, positive: 0, negative:0}, Shopping: {neutral: 0,
positive: 0, negative:0} }
  if (rereduce) {
    for(i=0; i<keys.length; i++) {
     if (keys[i] == "Melb") {
       switch(values[i].lifestyle) {
         case "Education":
           result.Education.neutral += values[i].neutral;
           result.Education.positive += values[i].positive;
           result.Education.negative += values[i].negative;
```

```
            break;
          case "Shopping":
            result.Shopping.neutral += values[i].neutral;
            result.Shopping.positive += values[i].positive;
            result.Shopping.negative += values[i].negative;
            break;
        }
      }
      if (keys[i] == "Syd") {
        switch(values[i].lifestyle) {
          case "Education":
            result_syd.Education.neutral += values[i].neutral;
            result_syd.Education.positive += values[i].positive;
            result_syd.Education.negative += values[i].negative;
            break;
          case "Shopping":
            result_syd.Shopping.neutral += values[i].neutral;
            result_syd.Shopping.positive += values[i].positive;
            result_syd.Shopping.negative += values[i].negative;
            break;
        }
      }
    }
    var output = [result, result_syd];
    return output;
  } else {
    return false;
  }
}
```