1. Consider the following code:

1: a = 7;
2: b = 10;
3: c = 20;
L1: 4: if (a <= c) goto L3;
5: d = 3*a;
6: if (d <= c) goto L2;
7: c = b + 11;
8: e = b + 10;
L2: 9: a = a + 1;
10: goto L1;
L3: 11: //a, b, c, d and e are all live after this

(a) What are the basic blocks for this code?
(b) Draw the block-level CFG for this code.
(c) What are the loop headers?
(d) What are the back edges (give the source and target lines of code)
(e) What are the loop-invariant instructions in this code?
(f) Which instructions can be moved out of the loop?
(g) What are the basic induction variables for the loops in this program? The mutual induction variables?
(h) Show the code after applying strength reduction
(i) Show the code after applying linear test replacement

2. Cam Piler thinks that loop interchange is always a good idea. Prove him wrong in two ways:

(a) Show that loop interchange is not always correct by giving an example of a nested loop where loop interchange is not legal.
(b) Describe a situation where loop interchange is legal, but is not actually useful (and may even be harmful).