1. (a) Consider a for loop statement with the following syntax:

$$\textsf{foreach}\ (\textsf{v in}\ E)\ S$$

The expression E must evaluate to an array. The foreach statement executes the statement S once for each element of the array, with the variable v bound to the array element at index $i-1$ on iteration i. Write the translation $\mathrm{T}[\![\ \textsf{foreach}\ (\textsf{v in}\ E)\ S\ ]\!]$ of this construct to low IR. Assume there are three-address instructions for loading an array element and for reading the array length.

(b) Consider a language for boolean expression with the following syntax:

$$E\ ::=\ E\ \textsf{nand}\ E\ \mid\ E\ \textsf{nor}\ E\ \mid\ E \Rightarrow E\ \mid\ E \Leftrightarrow E\ \mid\ id$$

In the above productions, nand is the negation of the logical and, nor is the negation of the logical or, $\Rightarrow$ is the implication, and $\Leftrightarrow$ is the equivalence operator. The nand and nor operators are short-circuit operators.

Assume that the only low IR operators for boolean expressions are and, or, and not. Write the translation $\mathrm{T}[\![\ E, L_1, L_2\ ]\!]$ of the logical expression E to Low IR which transfers the control to L_1 if E evaluates to true, and to L_2 if E evaluates to false.

(c) Using the translation from part (b), write the low IR code for the following statement:

$$\textsf{if (((a nand b) nor c)} \Rightarrow \textsf{d)}\ \ \textsf{a = true;}$$

2. Consider the following program which counts the primes from 2 to n using the sieve method:

```
for (i = 2; i < n; i++)
    a[i] = true;

count = 0;
for (i = 2; i < sqrt(n); i++)
    if (a[i]) {
        count = count + 1;
        for (j = 2*i; j < n; j = j+i)
            a[j] = false;
    }
```

In this program, a is an array of integers, and sqrt is a function which computes the square root. At the end of this program, only variable count is live.

(a) Write the three-address code (low IR) for this program. In the generated code, array indices represent the byte offsets (not element offsets). This means that you must translate statements such as a[j] = false into sequences t = i*4; a[t] = false.

(b) Construct the control flow graph from the low IR code.

(c) Show the dominator tree, the back edges, and the natural loops for the program in part (b).

(d) Optimize the program. Perform loop invariant code motion, strength reduction, and elimination of induction variables. Indicate the changes in the code for each optimization and show the final, optimized program.

3. We want to design a dataflow analysis which compute ranges for integer variables in the program. For this, we extend the set $\mathbb{N}$ of integer numbers with plus and minus infinity: $\mathbb{N}^* = \mathbb{N} \cup \{+\infty, -\infty\}$, such that $-\infty < n$, and $n < +\infty$ for any integer number n. We then use a lattice over the set $L = \{[l, u] \mid l, u \in \mathbb{N}^* \wedge l \leq u\}$. Every range of the form $[n, n]$ is a subrange of any other range in L that includes n.

   (a) Is this a forward or a backward analysis? Is this a may or a must analysis?

   (b) Define the partial order and the meet operator for this lattice.

   (c) Using this lattice to compute ranges of variables will fail. Explain why.

   (d) Consider a different lattice $L' = \{[l, u] \mid l, u \in \{-\infty, -1, 0, 1, +\infty\} \wedge l <= u\} \cup \{\top\}$. Draw the Hasse diagram for this lattice. Explain why we can successfully use this lattice for our analysis.

   (e) Assuming we use lattice L', show the transfer functions for statements of the form `x = y op z`, `x = y op c`, `x = -y`, and `x = y`, where `op` is an operator in the set {+, -, *} and `c` is a constant.

   (f) Using the lattice from part (c) and the transfer function from part (d), show how the dataflow analysis works for the following program:

```
x = 0;
while (...) {
    y = x;
    if (...) {
        x = x+1;
    } else {
        y = y-1;
    }
}
```