1. (a) The Loop has 10 instructions and 64 iterations of the loop are executed. Clearly, the number of instructions executed in course of program execution = (64 × 10) + 1 (for the first instruction of program before the loop) = **641**

   (b) Assume that vectors A, B and C are stored in main memory, and their addresses are in registers $t0, $t1 and $t2, respectively.

```
# i=0
lw    $t3, 0($t1)       # $t3 = b[0]
lw    $t4, 0($t2)       # $t4 = c[0]
add   $t4, $t3, $t4     # $t4 = b[0] + c[0]
sw    $t4, 0($t0)       # a[0] = b[0] + c[0]

# i=1
lw    $t3, 4($t1)       # $t3 = b[1]
lw    $t4, 4($t2)       # $t4 = c[1]
add   $t4, $t3, $t4     # $t4 = b[1] + c[1]
sw    $t4, 4($t0)       # a[1] = b[1] + c[1]
.........
.........
.........
# i=63
lw    $t3, 252($t1)     # $t3 = b[63]
lw    $t4, 252($t2)     # $t4 = c[63]
add   $t4, $t3, $t4     # $t4 = b[63] + c[63]
sw    $t4, 252($t0)     # a[63] = b[63] + c[63]
```

   This pattern would be repeated for all values of i < 64. Notice how loop unrolling gets rid of the branch maintainance code, instructions such as `slti`, `bne` and the overhead of maintaining the loop index is also removed.

   (c) Each iteration of the loop would require writing 4 instructions as shown above. So the length of entire unrolled code = (4 × 64) = **256 instructions**.

   (d) Loop code that has been unrolled by a factor of 2:

```
        add    $t4, $zero, $zero   # I1  i is initialized to 0, $t4 = 0
Loop:
                                   # UPDATE ADDRESS REGISTERS
        add    $a1, $t4, $t1       # I2  temp reg $a1 = address of b[i]
        add    $a2, $t4, $t2       # I3  temp reg $a2 = address of c[i]
        add    $a0, $t4, $t0       # I4  temp reg $a0 = address of a[i]
                                   # FIRST ADDITION
        lw     $t6, 0($a1)         # I5  temp reg $t6 = b[i]
        lw     $t7, 0($a2)         # I6  temp reg $t7 = c[i]
        add    $t6, $t6, $t7       # I7  temp reg $t6 = b[i] + c[i]
        sw     $t6, 0($a0)         # I8  a[i] = b[i] + c[i]
                                   # SECOND ADDITION
        lw     $t6, 4($a1)         # I9  temp reg $t6 = b[i + 1]
        lw     $t7, 4($a2)         # I10 temp reg $t7 = c[i + 1]
        add    $t6, $t6, $t7       # I11 temp reg $t6 = b[i + 1] + c[i + 1]
        sw     $t6, 4($a0)         # I12 a[i + 1] = b[i + 1] + c[i + 1]
```

```
                                # LOOP CODE
addi    $t4, $t4, 8             # I13 i = i + 8
slti    $t5, $t4, 256           # I14 $t5 = 1 if $t4 < 256, i.e. i < 64
bne     $t5, $zero, Loop        # I15 go to Loop if i < 256
```

(e) Each unrolling adds 4 instructions (2 loads, an add, and a store) to the static program:
static program size = 11 instructions + (4 instructions × (unrolling_factor - 1))

Each unrolling halves the number of overhead instructions executed:
number of instructions executed = $1 + (4 \times 64) + (\frac{6 \times 64}{\text{unrolling\_factor}})$