Question 5 (30 points):

Write the RISC-V assembly code for function `reverseAll`. The parameters for `reverseAll` are as follows:

- `a0`: the address of the first position of an array `S` of pointers to strings
- `a1`: the address of the first position of an array `L` of non-negative integers. For any index `i`, `L[i]` contains the length of the string whose address is in `S[i]`. The length of a string does not include the null character that terminates the string.

A position of the array S with the value `0xFFFFFFFF` is a sentinel that indicates the end of the array.

For each string whose address is in the array S, `reverseAll` must invoke the function `reverseString` from question 5 to reverse the order of the characters in the string.

`reverseAll` does not have any return values.

Your RISC-V code must follow all the register saving/restoring convention of RISC-V.

```
# reverseAll
# a0: address of first position of an array S.
#     Each element in S contains a pointer to a string.
# a1: address op first position of an array L. Each element L[i]
#     contains the length of the string whose pointer is in S[i].
# Pseudo code:
# ps <- address of first string pointer
# pl <- address of first length
# while(*ps != 0xFFFFFFFF)
#      reverseString(*ps, *pl)
#      ps++
#      pl++
reverseAll:
    addi sp, sp, -16
    sw s0, 0(sp)
    sw s1, 4(sp)
    sw s2, 8(sp)
    sw ra, 12(sp)
    mv s0, a0           # s0 <- initial string pointer
    mv s1, a1           # s1 <- initial lenght pointer
    li s2, -1           # s2 <- 0xFFFFFFFF
    lw a0, 0(s0)        # a0 <- *ps
    beq a0, s2, doneAll # if sentinel we are done
nextString:
    lw a1, 0(s1)        # a1 <- *pl
    jal reverseString   # reverseString
    addi s0, s0, 4      # ps++
    addi s1, s1, 4      # pl++
    lw a0, 0(s0)        # a0 <- *ps
    bne a0, s2, nextString # if not sentinel, go back
doneAll:
    lw s0, 0(sp)
    lw s1, 4(sp)
    lw s2, 8(sp)
    lw ra, 12(sp)
    addi sp, sp, 16
    jalr zero, ra, 0
```

Figure 1: A solution for `reverseString`.