Question 5 (30 points):

Assume that someone has implemented the function `IntToASCII` in RISC-V assembly that writes the ASCII representation of the value of an integer, followed by a single space, into a string. This function allocates the memory needed for the string.
You do not need to write code for this function.

- Input Parameter:
  - `a0`: an unsigned integer value `k`
- Output Value:
  - `a0`: the address of the first character of a null-terminated string containing the digits of `k` followed by a single space character.

In RARS the `PrintString` environment call can be used to print a string. The call number is `4`. In order to execute this call, your program must put the constant `4` in `a7` and the address of the null-terminated string in `a0`, and then issue the pseudo-instruction `ecall`.

In this question you are asked to write the function `PrintPalindromes` that:

a. prints all the palindromes in an interval `[s,e]`, where `s` is the first integer in the interval and `e` is the last integer in the interval.

b. returns the largest number that is a palindrome in the interval. If there are no palindromes in the interval, it returns the value 0xFFFF FFFF.

Your `PrintPalindromes` function must call the `Reverse` function and the `IntToASCII` function specified above. It must also use the RARS `PrintString` environment call to print the strings returned by the `IntToASCII` function.

- Input Parameter:
  - `a0`: an unsigned integer value `s`
  - `a1`: an unsigned integer value `e`
  - Guarantee: assume that `s` $\leq$ `e` for all calls of this function
- Output Parameter:
  - `a0`: the largest number that is a palindrome in the specified interval; or `0xFFFF FFFF` if there are no palindromes in the interval.

```
PrintPalindromes:
    addi    sp, sp, -16
    sw      s0, 0(sp)
    sw      s1, 4(sp)
    sw      s2, 8(sp)
    sw      ra, 12(sp)
    mv      s0, a0          # i <- s
    mv      s1, a1          # e
    li      s2, -1          # max <-- -1
loop:
    blt     s1, s0, done    # if e < s goto done
    mv      a0, s0          # i
    jal     Reverse         # r <- Reverse(i)
    bne     a0, s0 increment# if i != r goto increment
    mv      s2, a0          # max <- i
    jal     InttoASCII      # a0 <- address of allocated string
    li      a7, 4
    ecall                   # call PrintString
increment:
    addi    s0, s0, 1       # i++
    j       loop
done:
    mv      a0, s2          # return max
    lw      s0, 0(sp)
    lw      s1, 4(sp)
    lw      s2, 8(sp)
    lw      ra, 12(sp)
    addi    sp, sp, 16
    ret
```

Figure 1: A solution for `PrintPalindromes`.