A non-negative integer number is a palindrome if the number can be read from the left to the right or from the right to the left and it has the same value. For example, the numbers 0, 1, 2, 3, ... , 8, 9, 11, 22, 33 ..., 99, 101, 121, ... 1001, 1221, ... 14341, 91719, are examples of integer-number palindromes.

A known algorithm to determine if a number `n` is a palindrome consists in computing `r`, the reverse of `n`, and then comparing the values of `n` and `r`, if they are identical than the number is a palindrome. The following is a known algorithm[1] to compute the value of `r` for a given value of `n`:

```
reverse(n):
      r = 0
      while(n > 0)
             {
              r = r*10 + n%10
              n = floor(n/10)
             }
       return r
```

where `n%10` is the remainder of the division of `n` by `10`, and `floor(n/10)` is the floor of the division of `n` by `10`. For example, if `n=147`, the algorithm above will execute as follows:

```
n = 147
r = 0
r = 0*10 + 147%10 = 0 + 7 = 7
n = floor(147/10) = 14
r = 7*10 + 14%10 = 70 + 4 = 74
n = floor(14/10) = 1
r = 74*10 + 1%10 = 740 + 1 = 741
n = floor(1/10) = 0
```

Recall that RISC-V has integer division instructions for signed (DIV) and for unsigned (DIVU) numbers. It also has instructions to compute the remainder of the division for signed (REM) and unsigned (REMU) integers. For instance:

```
DIVU t0, t1, t2       # t0 <-- floor(t1,t2)
REMU s0, s1, s2       # s0 <-- s1%s2
```

One restriction is that the destination register for the `DIVU` instruction cannot be the same as one of the operands. For instance, `DIVU t0, t0, t1` is not a legal instruction.

The assembly code that you write for both parts of this question must follow all the register saving/restoring conventions for RISC-V.

[1] `https://www.geeksforgeeks.org/program-to-check-the-number-is-palindrome-or-not/`

Question 5 (20 points): Write RISC-V assembly for the function `Reverse` that computes the reverse of the value of an unsigned integer. This function has a single argument:

- `a0`: value of an unsigned number `n`

`Reverse` has a single return value:

- `a0`: the reverse value of the unsigned number `n`

```
Reverse:
    mv   t0, a0          # n
    mv   a0, zero        # r <- 0
    li   t1, 10          # t1 <- 10
nextDigit:
    ble  t0, zero, Rev_ret  # if n <= zero goto Rev_ret
    mul  t3, a0, t1      # t3 <- r*10
    remu t4, t0, t1      # t4 <- n%10
    add  a0  t3, t4      # r <- r*10 + n%10
    divu t5, t0, t1      # t5 <- floor(n/10)
    mv   t0, t5          # n <- floor(n/10)
    j    nextDigit
Rev_ret:
    ret
```

Figure 1: A solution for `Reverse`.