Arquitectura de Computadores I – Teste Prático de Aferição

Nome:____ Nº Mecanográfico: _________

Codifique em Assembly do MIPS o programa seguinte. Considere os protótipos das funções seguintes:

```
int countMin(char, char, char *);
int testlimit(char *, char, char);

int main(void)
{
    static char frase[15];
    int k = 0;                              // Use $S2

    print_str("Digite uma frase:\n");
    read_str(frase, 15);
    k = countMin('a', 'z', frase);
    print_str("Numero de minusculas: ");
    print_int10(k);
    return 0;
}
```

```
        .data
str1:   .asciiz "Digite uma frase:\n"
str2:   .asciiz "Numero de minusculas: "
frase:  .space 15

        .text
        .globl main
main:                                   # int main(void)
        addiu   $sp, $sp, -8            # {
        sw      $ra, 0($sp)
        sw      $s2, 4($sp)
        li      $s2, 0                  #    int k = 0;
        li      $v0, 4
        la      $a0, str1
        syscall                         #    print_str (str1);
        la      $a0, frase
        li      $a1, 15
        li      $v0, 8
        syscall                         #    read_str(frase, 15);
        li      $a0, 'a'
        li      $a1, 'z'
        la      $a2, frase
        jal     countMin                #
        move    $s2, $v0                #    k = countMin('a', 'z', frase);
        li      $v0, 4
        la      $a0, str2
        syscall                         #    print_str (str2);
        move    $a0, $s2
        li      $v0, 1
        syscall                         #    print_int10(k);
        lw      $s2, 4($sp)
        lw      $ra, 0($sp)
        addiu   $sp, $sp, 8
        li      $v0, 0                  #    return 0;
        jr      $ra                     # }
```

```c
int countMin(char minv, char maxv, char *arr)
{
     int nelem = 0;                                         // Use $s0
     int i;                                                 // Use $s1
     for (i = 0; arr[i] > '\0'; i++)
     {
           if (testlimit(&(arr[i]), maxv, minv) == 1) nelem++;
     }
     return nelen;
}
```

```
countMin:   addiu   $sp, $sp, -24          # int count(int *arr, int max, int count) {
            sw      $ra, 0($sp)
            sw      $s0, 4($sp)
            sw      $s1, 8($sp)
            sw      $s2, 12($sp)
            sw      $s3, 16($sp)
            sw      $s4, 20($sp)
            li      $s0, 0                 #    int nelem = 0;
            move    $s2, $a0               #    char ch1 = minv;
            move    $s3, $a1               #    char ch2 = maxv;
            move    $s4, $a2               #    char* cpt = arr;
            li      $s1, 0                 #    int i = 0;
countMin_for:
            addu    $t0, $s4, $s1          #     while (arr[i] > '\0')
            lb      $t1, 0($t0)            #    {
            ble     $t1, $0, countMin_forend
            move    $a0, $t0
            move    $a1, $s3
            move    $a2, $s2
            jal     testlimit
            bneq    $v0, 1, countMin_endif #          if (testlimit(&(arr[i]),maxv,minv)== 1)
            addi    $s0, $s0, 1            #               nelem++;
countMin_endif:
            addi    $s1, $s1, 1            #          i++;
            j       countMin_for           #     }
countMin_forend:
            move    $v0, $s0               #    return nelem
            lw      $ra, 0($sp)
            lw      $s0, 4($sp)
            lw      $s1, 8($sp)
            lw      $s2, 12($sp)
            lw      $s3, 16($sp)
            lw      $s4, 20($sp)
            addiu   $sp, $sp, 24
            jr      $ra                    # }
```

```c
int testlimit(char *ch, char max, char min)
{
     int inside = 1;                                        // Use $t0
     if (*ch < min || *ch > max) inside = 0;
     max = 0;
     min = 0;
     return inside;
}
```

```
testlimit:                                   # int testlimit(char *ch, char max, char min){
            li      $t0, 1                   #    int inside = 1;
            lb      $t1, 0($a0)
            bge     $t1, $a2, testlimit_next #    if (*ch < min) || *ch > max)
            j       t_reset                  #    {
testlimit_next:
            ble     $t1, $a1, testlimit_next1#
t_reset:    li      $t0, 0                   #        inside = 0;
testlimit_next1:
            move    $v0, $t0                 #    return inside;
            jr      $ra                      # }
```