SWAP TWO CONSECUTIVE CHARS IN STRING

```
        .data
prompt: .asciz "Enter string in: "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        # Rejestry
        la t0, buf
        la t1, buf
        li t4, '\n'

        addi t1, t1, 1

        lb t2, (t0)
        lb t3, (t1)

        beqz t3, end

swap:
        lb t2, (t0)
        lb t3, (t1)
        beq t2, t4, end
        beq t3, t4, end
        beqz t3, end
        sb t3, (t0)
        sb t2, (t1)
        addi t0, t0, 2
        addi t1, t1, 2
        j swap

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

Determine String Length

```
        .data
prompt: .asciz "Enter your string: "
prompt2: .asciz "String lenghth: "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf              # Wskaźnik na początek bufora
        lb t1, (t0)             # Wczytaj znak
        li t2, 0            # Licznik znaków

        beqz t1, end            # Zakończ jeśli pusty string

loop:
        lb t1, 0(t0)            # Wczytaj znak
        beqz t1, end            # Jak napotkasz '\0' zakończ
        addi t0, t0, 1
        addi t2, t2, 1
        j loop

end:
        addi t2, t2, -1         # Usuń znak końcowy

        li a7, 4
        la a0, prompt2
        ecall

        li a7, 1
        mv a0, t2
        ecall

        li a7, 10
        ecall
```

Reverse string

```
        .data
prompt: .asciz "Enter string in: "
buf: .space 100
        .text
        .global main

main:
        li a7, 4
        la a0 prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf              # Pierwszy wskaźnik na początek bufora
        la t1, buf              # Drugi wskaźnik który będzie wskaźnikiem na koniec bufora
        li t2, 0        # Char counter

        lb t3, (t0)
        beqz t3, end

count_chars:
        lb t3, (t0)
        beqz t3, increment_buf_pointer
        addi t0, t0, 1
        addi t2, t2, 1
        j count_chars

increment_buf_pointer:
        la t0, buf
        add t1, t1, t2
        addi t1, t1, -1

loop:
        ble t5, t4, end
        lb t4, (t0)             # Pierwszy znak
        lb t5, (t1)             # Ostatni znak
        sb t5, (t0)
        sb t4, (t1)
        addi t0, t0, 1
        addi t1, t1, -1
        j loop

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

REVERSE THE ORDER IN STRING

```
                .data
prompt: .asciz "Enter string in: "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        # Rejestry
        la t0, buf
        la t1, buf
        li t3, '\n'

        lb t4, (t0)
        beqz t4, end

count_string:
        lb t4, (t1)
        beq t4, t3, delete_whitespace
        addi t1, t1, 1
        j count_string

delete_whitespace:
        addi t1, t1, -1

loop:
        lb t4, (t0)
        lb t2, (t1)
        ble t1, t0, end
        sb t4, (t1)
        sb t2, (t0)
        addi t0, t0, 1
        addi t1, t1, -1
        j loop

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

Replace each character belonging to a word by the length of the word (mod 10).

```
        .data
prompt: .asciz "Enter string in :"
buf1: .space 100
buf2: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf1
        li a1, 100
        ecall

        la t0, buf1             # Pointer bufora pierwszego
        la t1, buf2             # Pointer bufora drugiego
        li t2, 32
        li t3, 10
        li s0, 0        # Char counter
        li s6, 10

size_of:
        lb t4, (t0)
        addi t0, t0, 1
        beq t4, t2, write
        beq t4, t3, write
        addi s0, s0, 1
        j size_of

write:
        sb s0, (t1)
        addi t1, t1, 1
        li s0, 0
        beq t4, t3, part2
        j size

part2:
        la t0, buf1
        la t1, buf2

loop:
        lbu t4, (t0)
        lbu t5, (t1)
        rem t5, t5, t6
        beq t4, t2, nowrt
        beq t4, t3, nowrt
```

```
        addi t5, t5, 48
        sb t5, (t0)
        addi t0, t0, 1
        j loop

nowrt:
        beq t4, t3, end
        addi t0, t0, 1
        addi t1, t1, 1
        j loop

end:
        li a7, 4
        la a0, buf1
        ecall

        li a7, 10
        ecall
```

Replace each character belonging to a word by the length of the word (mod 10).

Write function remove which removes from the source string every small letter. remove returns the length of
the resulting string.
Source> Computer Architecture Lab
Result> C A L
Return value: 5

```
        .data
prompt1: .asciz "Source> "
prompt2: .asciz "Result> "
prompt3: .asciz "Return value: "
newLine: .ascii "\n"
buf1:   .space 100
buf2:   .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt1
        ecall

        li a7, 8
        la a0, buf1
        li a1, 100
        ecall

        la t0, buf1
        la t1, buf2
        li t2, 'A'
        li t3, 'Z'
        li t4, ' '

        li s0, 0            # Char count
        li s1, 0

        lb t5, (t0)
        beqz t5, end

seek_for_uppercase:
        lb t5, (t0)
        beqz t5, end_string
        blt t5, t2, next_char
```

```
        bgt t5, t3, next_char
        sb t5, (t1)
        addi t1, t1, 1
        sb t4, (t1)
        addi t1, t1, 1

next_char:
        addi t0, t0, 1
        j seek_for_uppercase

end_string:
        sb zero, (t1)

        li s0, 0
        li s1, 0
        la t1, buf2
count_chars_buf2:
        lb s0, (t1)
        beqz s0, print_chars
        addi t1, t1, 1
        addi s1, s1, 1
        j count_chars_buf2

print_chars:
        addi s1, s1, -1

end:
        li a7, 4
        la a0, prompt2
        ecall

        li a7, 4
        la a0, buf2
        ecall

        li a7, 4
        la a0, newLine
        ecall

        li a7, 4
        la a0, prompt3
        ecall

        li a7, 1
        mv a0, s1
        ecall

        li a7, 10
        ecall
# Write function remove which removes from the source string every small letter. remove returns the length of
# the resulting string.
# Source> Computer Architecture Lab
# Result> C A L
# Return value: 5
```

Filter Ints from string

```
                .data
prompt1: .asciz "Source> "
prompt2: .asciz "Result> "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt1
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf
        li t1, '0'
        li t2, '9'

        lb t3, (t0)
        beqz t3, end

loop:
        lb t3, (t0)
        beqz t3, end
        blt t3, t1, next
        bgt t3, t2, next

delete:
        lb t3, (t0)
        addi t0, t0, 1
        lb t3, (t0)
        addi t0, t0, -1
        sb t3, (t0)
        addi t0, t0, 1
        bnez t3, delete

        la t0, buf
        lb t4, (t0)
        addi t0, t0, -1

next:
        addi t0, t0, 1
        j loop

end:
        li a7, 4
        la a0, prompt2
        …
```

The first and the second character in the string represent the (begin and the end) markers, which define a
substring. Your task is to replace all characters between the first occurrence of begin marker and first
occurcence of the end marker with * character. If there is no begin or end marker in the input string (the string
after the : character), then nothing should be changed. Replace the first three characters of the string with
spaces.
Input string > oi:wind on the hill
Conversion results> wind *********ll

```
        .data
prompt1: .asciz "Input string: "
prompt2: .asciz "Conversion results"
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt1
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf              # Wskaźnik na ciąg znaków
        li t1, 0        # Rejestr na pierwszą literę
        li t2, 0        # Rejestr na drugą literę
        li t3, ':'
        li t4, ' '
        li s0, '*'
        li s1, 0        # Rejestr na długość stringa

        lb t5, (t0)
        beqz t5, end

seek_for_colon:
        lb t5, (t0)
        beq t5, t3, colon_found
        beqz t5, reset_buf_pointer
        addi t0, t0, 1
        addi s1, s1, 1
        j seek_for_colon
```

```
# Tu jest błąd, przesuń stringa o trzy miejsca w prawo po czym wpisz spacje w miejsca przed stringiem
reset_buf_pointer:
        la t0, buf
        add t0, t0, s1

insert_spaces:
        lb t5, (t0)
        beqz s1, end
        sb t4, (t0)
        addi s1, s1, -1
        j insert_spaces

colon_found:
        addi t0, t0, -2
        lb t1, (t0)
        sb t4, (t0)
        addi t0, t0, 1
        lb t2, (t0)
        sb t4, (t0)
        addi t0, t0, 1
        sb t4, (t0)
        addi t0, t0, 1          # Ustaw wskaźnik na początek dobrego stringa

transform_string:
        lb t5, (t0)
        beq t5, t1, put_asterisks
        beqz t5, end
        addi t0, t0, 1
        j transform_string

put_asterisks:
        lb t5, (t0)
        sb s0, (t0)
        beq t5, t2, end
        addi t0, t0, 1
        j put_asterisks

end:
        li a7, 4
        la a0, prompt2
        ecall

        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

Input string > oi:wind on the hill
Conversion results> wind *********ll

When [ ] then put asterisks in between

```
        .data
prompt: .asciz "Enter string in "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf                  # Buffor pointer
        li s0, 0            # Opening bracket pointer
        li s1, 0            # Closing bracket pointer

        li t1, '['
        li t2, ']'
        li t3, '*'

        lb t4, (t0)
        beqz t4, end

search_for_opening_bracket:
        lb t4, (t0)
        beqz t4, end
        beq t4, t1, opening_found
        addi t0, t0, 1
        j search_for_opening_bracket

opening_found:
        mv s0, t0
        addi s0, s0, 1

search_for_closing_bracket:
        lb t4, (t0)
        beqz t4, end
        beq t4, t2, closing_found
        addi t0, t0, 1
        j search_for_closing_bracket

closing_found:
        mv s1, t0
        la t0, buf

put_asterisks:
```

```
        lb t4, (s0)
        beqz t4, end
        beq s0, s1, end
        sb t3, (s0)
        addi s0, s0, 1
        j put_asterisks

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

When [ ] then put asterisks in between

At the beginning of the output string put the characters from the odd positions, next the even.

```
                .data
prompt: .asciz "Enter string in: "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        # Rejestry
        la t0, buf              # Wskaźnik znaków nieparzystych
        la t1, buf              # Wskaźnik znaków parzystych
        la t2, buf              # Wskaźnik wstawiający znaki
        li t5, '\n'

        lb t3, (t0)
        beqz t3, end

        addi t1, t1, 1

odd_numbers:
        lb t3, (t0)
        beqz t3, even_numbers
        beq t3, t5, even_numbers
        sb t3, (t2)
        addi t0, t0, 2
        addi t2, t2, 1
        j odd_numbers

even_numbers:
        lb t4, (t1)
        beqz t4, end
        beq t4, t5, end
        sb t4, (t2)
        addi t1, t1, 2
        addi t2, t2, 1
        j even_numbers

end:    ……
```

CONVERT UPPERCASE LETTERS TO ASTERISKS

```
        .data
prompt: .asciz "Enter string in: "
buf: .space 100
        .text
        .global main
main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        li t0, 'A'
        li t1, 'Z'
        li t2, '*'

        la t3, buf

        lb t4, (t3)
        beqz t4, end

loop:
        lb t4, (t3)
        beqz t4, end
        blt t4, t0, next
        bgt t4, t1, next
        sb t2, (t3)
        addi t3, t3, 1
        j loop

next:
        addi t3, t3, 1
        j loop

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

CONVERT LOWERCASE LETTERS TO ASTERISKS

```
        .data
prompt: .asciz "Enter string: "
buf: .space 100
        .text
        .global main
main:
        li a7, 4
        la a0, prompt
        ecall

        li a7 ,8
        la a0, buf
        li a1, 100
        ecall

        li t0, 'a'
        li t1, 'z'
        li t2, '*'

        la t3, buf                  # wskaźnik na stringa

        lb t4, (t3)
        beqz t4, end

 loop:
        lb t4, (t3)
        beqz t4, end
        blt t4, t0, next
        bgt t4, t1, next
        sb t2, (t3)
        addi t3, t3, 1
        j loop

next:
        addi t3, t3, 1
        j loop

end:
        li a7, 4
        la a0, buf
        ecall

        li a7, 10
        ecall
```

CONVERT ALL NON LETTER SYMBOLS INTO ASTERISKS

```
        .data
prompt: .asciz "Enter string in: "
buf: .space 100

        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        # Rejestry
        li t0, 'A'
        li t1, 'Z'
        li t2, 'a'
        li t3, 'z'
        li t6, '*'

        # Wskaźnik na stringa
        la t4, buf

        lb t5, (t4)
        beqz t5, end

loop:
        lb t5, (t4)
        li s1, '\n'
        beq t5, s1, end
        blt t5, t0, asterisk
        bgt t5, t3, asterisk
        ble t5, t1, skip
        bge t5, t2, skip

asterisk:
        sb t6, (t4)

skip:
        addi t4, t4, 1
        j loop

end:
        …
```

Łączenie dwóch stringów w jeden

```
.data
prompt1: .asciz "Enter first string: "
prompt2: .asciz "Enter second string: "
prompt3: .asciz "Result: "
buf1: .space 100
buf2: .space 100
result_buf: .space 200

.text
.global main

main:
   # Wczytywanie pierwszego ciągu
   li a7, 4
   la a0, prompt1
   ecall

   li a7, 8
   la a0, buf1
   li a1, 100
   ecall

   # Wczytywanie drugiego ciągu
   li a7, 4
   la a0, prompt2
   ecall

   li a7, 8
   la a0, buf2
   li a1, 100
   ecall

   # Łączenie ciągów
   la t0, buf1          # Wskaźnik na buf1
   la t1, buf2          # Wskaźnik na buf2
   la t2, result_buf     # Wskaźnik na bufor wynikowy

   # Kopiowanie pierwszego ciągu do bufora wynikowego
   copy_first:
      lb t3, 0(t0)       # Wczytaj znak z buf1
      beqz t3, copy_second # Jeśli napotkano '\0', przejdź do kopiowania drugiego ciągu
      sb t3, 0(t2)       # Zapisz znak z buf1 do bufora wynikowego
      addi t0, t0, 1     # Przesuń wskaźnik na buf1
      addi t2, t2, 1     # Przesuń wskaźnik na bufor wynikowy
      j copy_first

   # Kopiowanie drugiego ciągu do bufora wynikowego
   copy_second:
      lb t3, 0(t1)       # Wczytaj znak z buf2
      beqz t3, end_copy   # Jeśli napotkano '\0', zakończ kopiowanie
      sb t3, 0(t2)       # Zapisz znak z buf2 do bufora wynikowego
      addi t1, t1, 1     # Przesuń wskaźnik na buf2
```

```
        addi t2, t2, 1      # Przesuń wskaźnik na bufor wynikowy
        j copy_second

    end_copy:

    # Wyświetlenie wyniku
    li a7, 4
    la a0, prompt3
    ecall

    # Wyświetlenie zawartości bufora wynikowego bez uwzględniania '\0'
    la t0, result_buf      # Wskaźnik na bufor wynikowy
print_result:
    lb a0, 0(t0)         # Wczytaj znak z bufora wynikowego
    beqz a0, end_print    # Jeśli napotkano '\0', zakończ wyświetlanie
    li a7, 11            # Wywołanie systemowe do wyświetlenia znaku
    ecall
    addi t0, t0, 1       # Przesuń wskaźnik na bufor wynikowy
    j print_result

end_print:

    # Zakończenie programu
    li a7, 10
    ecall
```

Łączenie dwóch stringów w jeden

Bubble Sort

```
        .data
prompt: .asciz "Enter string in: "
buf: .space 100
        .text
        .global main

main:
        li a7, 4
        la a0, prompt
        ecall

        li a7, 8
        la a0, buf
        li a1, 100
        ecall

        la t0, buf
        li t1, 0            # Licznik znaków
        li t2, '\n'

        lb t3, (t0)
        beqz t3, print_sorted_string

count_length:
        lb t3, (t0)       # Wczytaj kolejny znak
        beqz t3, sort      # Jeśli koniec ciągu, przejdź do sortowania
        beq t3, t2, sort   # Jeśli znak nowej linii, przejdź do sortowania
        addi t1, t1, 1     # Zwiększ licznik znaków
        addi t0, t0, 1     # Przesuń wskaźnik na następny znak
        j count_length     # Powtarzaj, dopóki nie skończy się ciąg

sort:
  # Pętla sortowania bąbelkowego
        li t4, 1          # Flaga wskazująca, czy nastąpiła zamiana
outer_loop:
        li t4, 0          # Zresetuj flagę zamiany
        la t0, buf        # Ustaw wskaźnik na początek bufora
inner_loop:
        lb t5, (t0)       # Wczytaj znak
        lb t6, 1(t0)      # Wczytaj następny znak

  # Jeśli t6 == '\n' || t6 == 0, to koniec ciągu, przejdź do końca sortowania
        beqz t6, end_sort
        beq t6, t2, end_sort

        blt t6, t5, swap    # Jeśli t6 < t5, zamień je
        addi t0, t0, 1      # Przesuń wskaźnik na następny znak
        j inner_loop        # Powtarzaj, dopóki nie skończy się ciąg

swap:
  # Zamień wartości t5 i t6
        sb t6, (t0)       # Zapisz t6 w miejscu t5
```

```
        sb t5, 1(t0)        # Zapisz t5 w miejscu t6
        li t4, 1            # Ustaw flagę zamiany na 1
inner_loop_end:
        addi t0, t0, 1      # Przesuń wskaźnik na następny znak
        j inner_loop        # Powtarzaj, dopóki nie skończy się ciąg

end_sort:
        beqz t4, print_sorted_string  # Jeśli nie było zamian, przejdź do wypisywania posortowanego ciągu
        li t4, 0            # Zresetuj flagę zamiany
        j outer_loop        # Powtarzaj zewnętrzną pętlę sortowania

print_sorted_string:
  # Wyświetl posortowany ciąg znaków
        li a7, 4
        la a0, buf
        ecall

  # Zakończ program
        li a7, 10
        ecall
```

Bubble Sort