Lab 3

Lab 2

Pseudocode

```
i = 0
j = 0
do
    c = str[i]
    if c != 32
       res[j] = c
       j += 1
    i += 1
while c != 0
```

Code in RISC-V

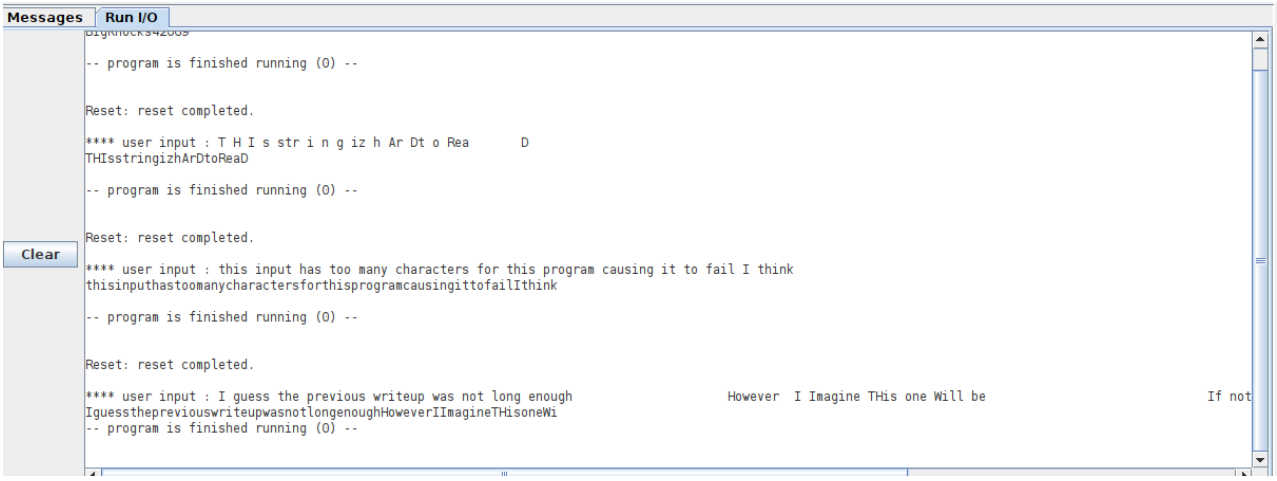
```
# TODO
    # remove spaces in str
    # print res
    addi t4, x0, 0 # i = 0
    addi t5, x0, 0 # j = 0
    addi a2, x0, 32 # len till stop
    loop:
        add t1, a0, t4 # go to str[0] + i in t1 (calculate offest)
        lb t6, 0(t1) # c = str[i] read byte at i in the string
        bne t6, a2, if # if c != 32, goto if
        beq x0, x0, ifexit
        if:
            add t1, a1, t5 # a1 + j in t1(calculate offset)
            sb t6, 0(t1) #res[j] = c
            addi t5, t5, 1 # j += 1
        ifexit:
            addi t4, t4, 1 #i += 1
            beq t6, zero, exitloop
            beq x0, x0, loop
    exitloop:
    li a7, 4 # Load print statment
    add a0, a1, zero #stage a0
```

```
    ecall                # make the call
```

I/O Results

- Testcase 1: 1 2 3 4 5 6
  -
    ```
    Reset: reset completed.

    **** user input : 1 2 3 4 5
    12345

    -- program is finished running (0) --
    ```
  - Results: Works on Numeric Inputs
- Testcase 2: Text Strings of varrying lengths
  - 

  - Results: Learned there is a max length of characters which can be entered as the string, I believe this has to do with RISC-V being 32 bit, or the page size

Analysis

The code seems to work for most inputs of characters, the only restrictions I've noticed were on string length. The code also seems to not be wasteful of any instructions, as you can see it will calculate the correct answer quickly. It has to go through every bit in the string before terminating.