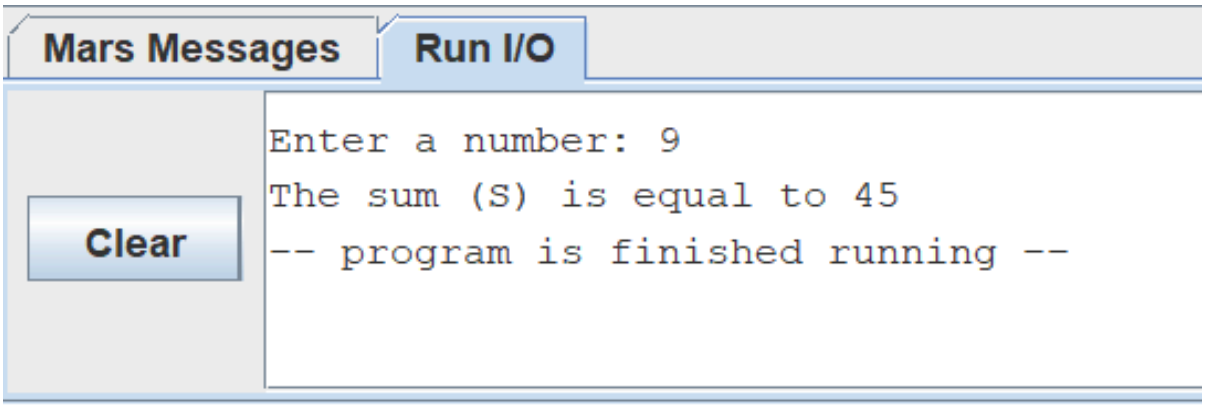
```
1: # create variables n, sum, count
2: # create a prompt message for user input!
3: .data
4: n: .word 0
5: count: .word 0
6: sum: .word 0
7: prompt1: .asciiz "Enter a number: "
8: prompt2: .asciiz "The sum (S) is equal to "
9:
10: .text
11: .globl main
12: main:
13: # print prompt
14: li $v0, 4
15: la $a0, prompt1
16: syscall
17:
18: # read integer and store in 'n'
19: li $v0, 5
20: syscall
21: sw $v0, n
22:
23: # initialize count, sum to 0
24: li $t0, 0
25: li $t1, 0
26:
27: # load 'n' into a register
28: lw $t2, n
29:
30: # run a loop
31: Loop:
32:     bge $t0, $t2, exit
33:     addi $t0, $t0, 1
34:     add, $t1, $t1, $t0
35:
36:     j Loop
37: exit:
38:     # store sum back in memory
39:     sw $t1, sum
40:
41:     # print sum w/ message
42:     li $v0, 4
43:     la $a0, prompt2
44:     syscall
45:     li $v0, 1
46:     move $a0, $t1
47:     syscall
48:
49:     # indicate end of program
50:     li $v0, 10
51:     syscall
```

Vedaa Anand
1020221201
Computer Organization and Systems, Spring 2025

Output

Question 1



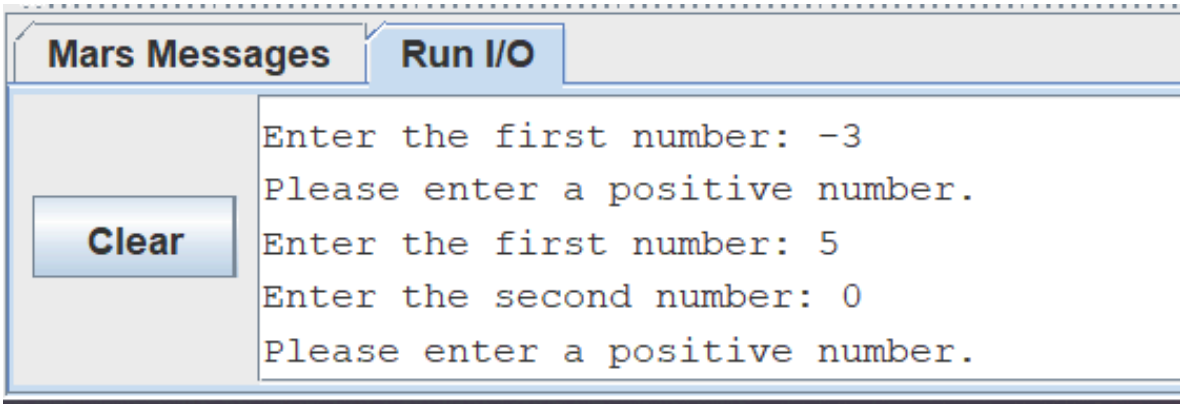
```
1: # create 2 prompts
2: .data
3: x: .word 0
4: y: .word 0
5: prompt1: .asciiz "Enter the first number: "
6: prompt2: .asciiz "Enter the second number: "
7: prompt3: .asciiz "Please enter a positive number.\n"
8: prompt4: .asciiz "The GCD is: "
9:
10: .text
11: .globl main
12: main:
13: # get first user input
14: li $v0, 4
15: la $a0, prompt1
16: syscall
17:
18: # read value
19: li $v0, 5
20: syscall
21: sw $v0, x
22: lw $t0, x
23:
24: # check if the inputted number is positive
25: Loop1:
26:     bge $t0, 1, exit1
27:     li $v0, 4
28:     la $a0, prompt3
29:     syscall
30:
31:     li $v0, 4
32:     la $a0, prompt1
33:     syscall
34:
35:     li $v0, 5
36:     syscall
37:     sw $v0, x
38:     lw $t0, x
39:
40:     j Loop1
41: exit1:
42:
43: # do the same for the next number
44: # get second user input
45: li $v0, 4
46: la $a0, prompt2
47: syscall
48:
49: # read value
50: li $v0, 5
51: syscall
```

```
52: sw $v0, y
53: lw $t1, y
54:
55: Loop2:
56:     bge $t1, 1, exit2
57:     li $v0, 4
58:     la $a0, prompt3
59:     syscall
60:
61:     li $v0, 4
62:     la $a0, prompt2
63:     syscall
64:
65:     li $v0, 5
66:     syscall
67:     sw $v0, y
68:     lw $t1, y
69:
70:     j Loop2
71: exit2:
72:
73: # gcd calculation
74: Loop3:
75:     # if x,y are equal just return x
76:     beq $t0, $t1, exit3
77:     # if x < y, let y = y -x
78:     blt $t0, $t1, if
79:     # if x > y, let x = x - y
80:     sub $t0, $t0, $t1
81:
82:     j Loop3
83: if:
84:     sub $t1, $t1, $t0
85:     j Loop3
86: exit3:
87:     # print prompt
88:     li $v0, 4
89:     la $a0, prompt4
90:     syscall
91:     # print the gcd
92:     li $v0, 1
93:     move $a0, $t0
94:     syscall
95:
96:     # end of program
97:     li $v0, 10
98:     syscall
99:
```

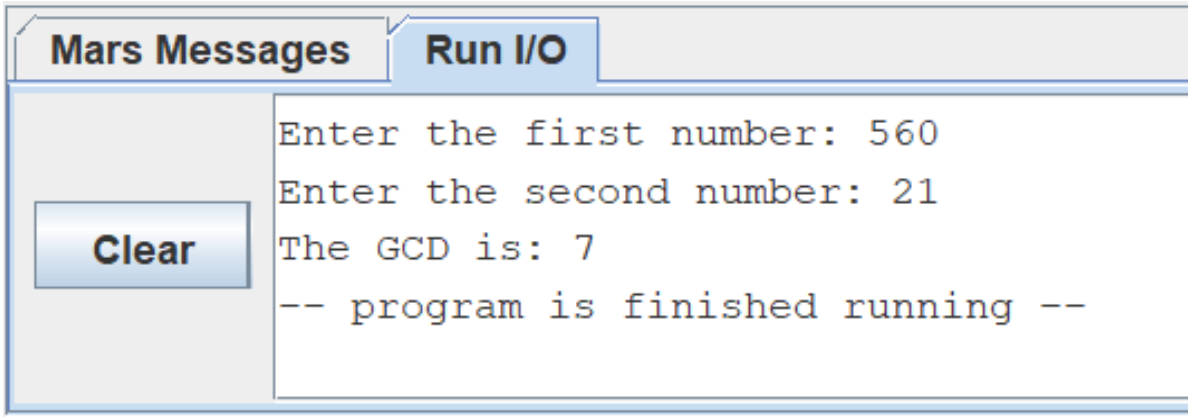
Output

Question 2

Check for positive input:



Correctness of program:



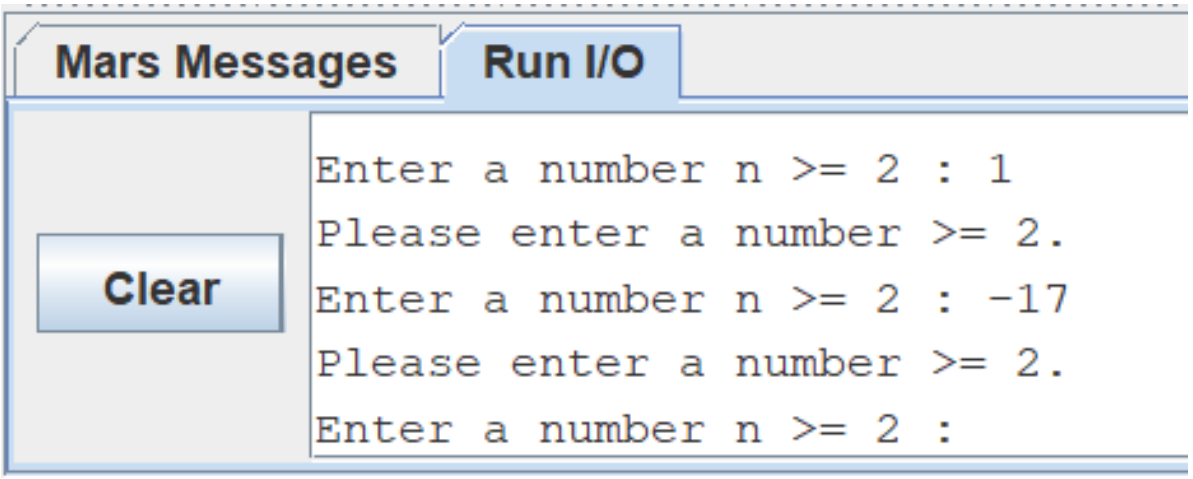
```
1: .data
2: initial_prompt: .asciiz "Enter a number n >= 2 : "
3: incorrect_input_prompt: .asciiz "Please enter a number >= 2.\n"
4: result_prompt: .asciiz "The nth Fibonacci number is "
5:
6: .text
7: .globl main
8: main:
9:  # ask prompt
10:     li $v0, 4
11:     la $a0, initial_prompt
12:     syscall
13:
14:     # get user input
15:     li $v0, 5
16:     syscall
17:     move $t0, $v0 # store n in $t0
18:
19:     # check if n < 2
20:     blt $t0, 2, invalid_input
21:
22:     # set variables x, y, counter
23:     li $t1, 0 # x = 0
24:     li $t2, 1 # y = 1
25:     li $t3, 2 # counter = 2, since fib(1) = 0, fib(2) = 1, etc
26:
27:     check_n:
28:         # check if counter < n
29:         bge $t3, $t0, return_result
30:     fib_loop:
31:         add $t4, $t1, $t2 # set z = x + y
32:         move $t1, $t2 # x = y
33:         move $t2, $t4 # y = z
34:
35:         addi $t3, $t3, 1 # counter++
36:         j check_n
37:
38:     return_result:
39:         li $v0, 4
40:         la $a0, result_prompt
41:         syscall
42:
43:         li $v0, 1
44:         move $a0, $t2
45:         syscall
46:
47:         li $v0, 10
48:         syscall
49:
50:     invalid_input:
51:         li $v0, 4
```

```
52:         la $a0, incorrect_input_prompt
53:         syscall
54:         j main
```

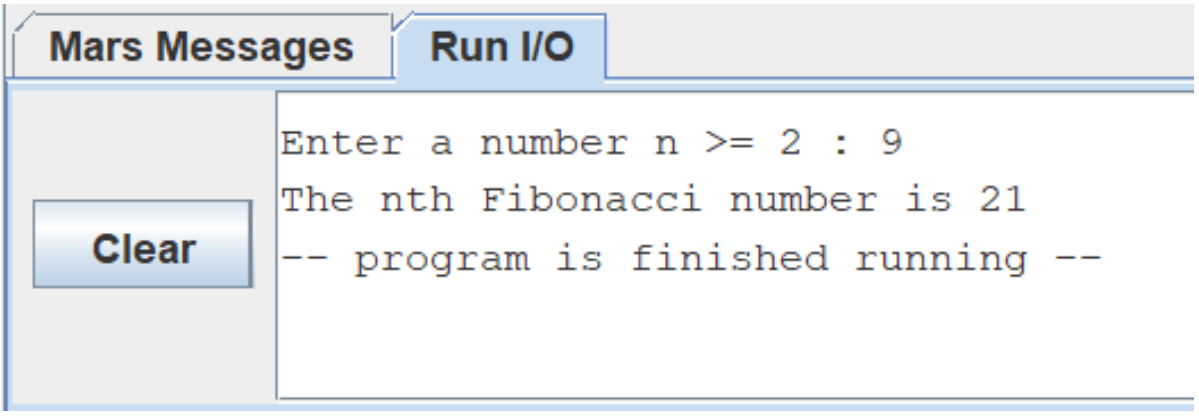
Output

Question 3:

Check for positive input (n >= 2):



Correctness of program:



```
1: .data
2: prompt: .asciiz "Enter an integer: "
3: error_prompt: .asciiz "Integer out of 32-bit range.\n"
4: hex_prefix: .asciiz "0x"
5: hex_digits: .asciiz "0123456789ABCDEF"
6: newline: .asciiz "\n"
7:
8: .text
9: .globl main
10: main:
11:     li $v0, 4
12:     la $a0, prompt
13:     syscall
14:
15:     li $v0, 5
16:     syscall
17:     move $t0, $v0
18:
19:     # load min, max values in registers
20:     li $t1, -2147483648
21:     li $t2, 2147483647
22:
23:     # check bounds
24:     blt $t0, $t1, invalid_input
25:     bgt $t0, $t2, invalid_input
26:
27:     # print the prefix
28:     li $v0, 4
29:     la $a0, hex_prefix
30:     syscall
31:
32:     # convert integer to hexa
33:     li $t1, 28 # shift, starting from first 4-bit chunk
34:
35:     hex_loop:
36:         # isolate chunk by shifting by $t1 bits
37:         srlv $t2, $t0, $t1
38:         andi $t2, $t2, 0xF # mask
39:
40:         # convert to ASCII: index and load
41:         la $t3, hex_digits
42:         add $t2, $t2, $t3
43:         lbu $a0, 0($t2)
44:
45:         # print
46:         li $v0, 11
47:         syscall
48:
49:         # shift to next
50:         subi $t1, $t1, 4
51:         bgez $t1, hex_loop
```

```
52:
53:         # print a new line
54:         li $v0, 4
55:         la $a0, newline
56:         syscall
57:
58:         # exit program
59:         li $v0, 10
60:         syscall
61:
62:     invalid_input:
63:         li $v0, 4
64:         la $a0, error_prompt
65:         syscall
66:
67:         j main
68:
69:
70:
```

Output

Question 4