COMPUTER ORGANIZATION PROJECT 1

REPORT

Team Members

Student Name / Student ID

Barış Giray Akman / 150121822
İrem Kıranmezar / 150121013
İrem Aydın / 150120013

Question 1

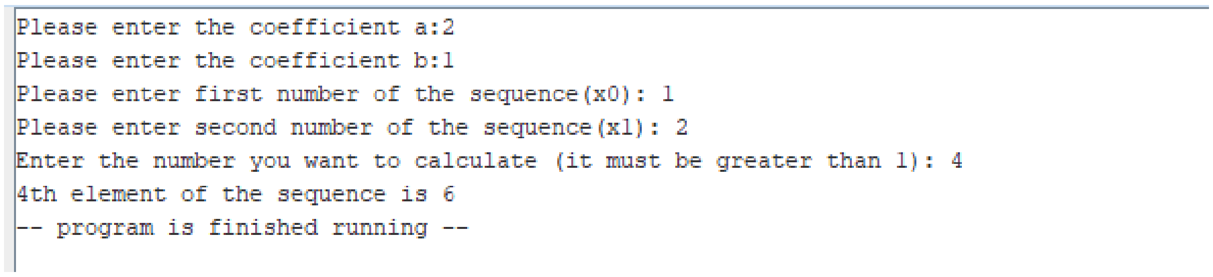
In the first problem, the objective is demanding the user to the enter inputs by an order and calculating the result according to the formula given in **Figure-1.**

$$f(x) = a * f(x-1) + b * f(x-2) - 2,$$

Figure-1: Formula given in the problem PDF

As an approach, program recursively stores the recent two values of f(x-1) and f(x-2). For the next iteration, it calculates the f(x). Then, the value of f(x-1) is assigned to f(x-2) and the value of f(x) is assigned to f(x-1).

During the execution of the program, some considerations must be followed. For instance, while getting the input through the user, each of them must be prompted one-by-one properly as it is given in the **Figure-2** below.

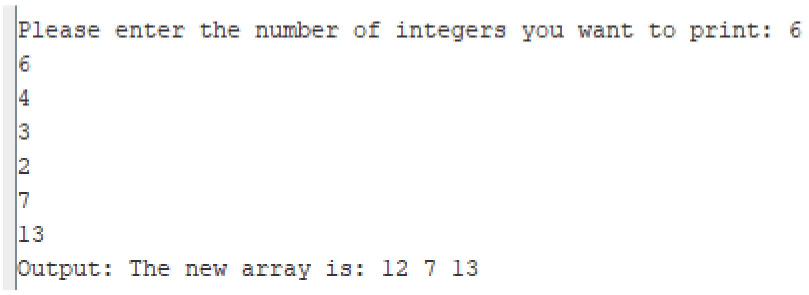


```
Please enter the coefficient a:2
Please enter the coefficient b:1
Please enter first number of the sequence(x0): 1
Please enter second number of the sequence(x1): 2
Enter the number you want to calculate (it must be greater than 1): 4
4th element of the sequence is 6
-- program is finished running --
```

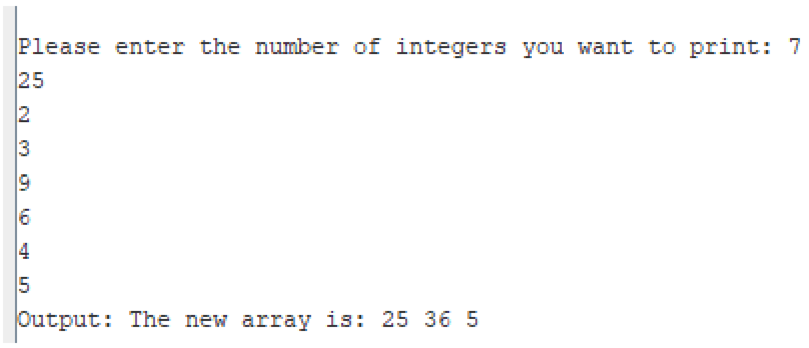
Figure-2: Sample execution of the program

Question 2

Aim of this question is traversing the array determined by the user and recursively finding the least common factor of two numbers and settling them to the array. Initially, in our program, number of integers that array will be filled must be entered at first. Then, integers must be typed one-by-one in underlined manner. Some sample cases are given below.

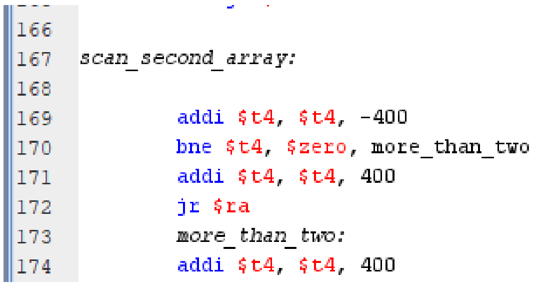


```
Please enter the number of integers you want to print: 6
6
4
3
2
7
13
Output: The new array is: 12 7 13
```

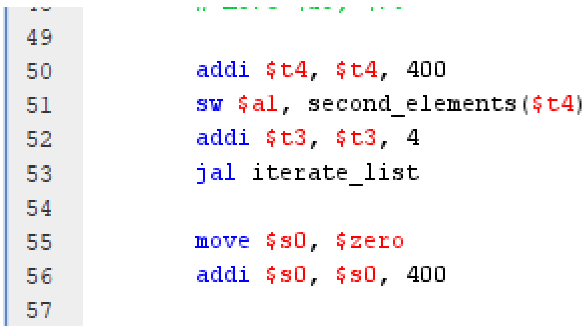


```
Please enter the number of integers you want to print: 7
25
2
3
9
6
4
5
Output: The new array is: 25 36 5
```

For the solution, two array has been utilized. Each of them's size are declared as 100. To change the size of the array, the lines below must also be changed too accordingly with array size. Because of each word holds 4 bytes, total byte of the array is 400 for the 100 number of array.



```

scan_second_array:

        addi $t4, $t4, -400
        bne $t4, $zero, more_than_two
        addi $t4, $t4, 400
        jr $ra
        more_than_two:
        addi $t4, $t4, 400
```



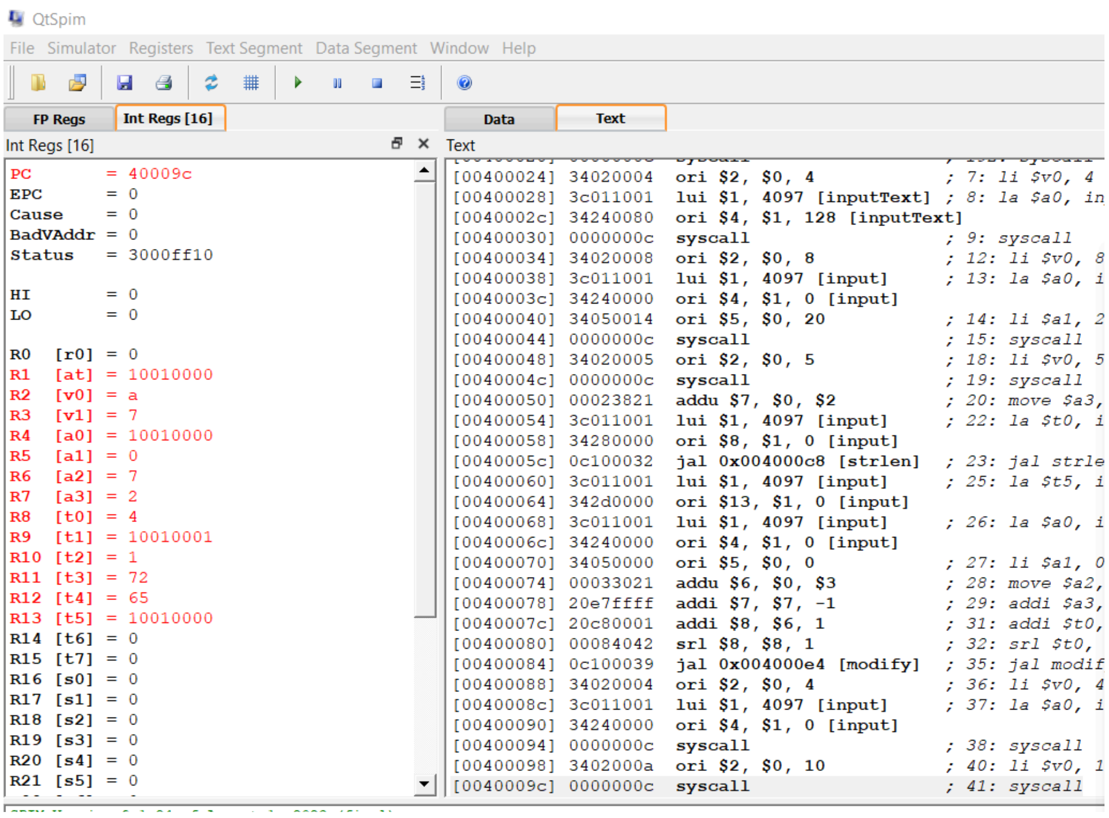
```

        addi $t4, $t4, 400
        sw $a1, second_elements($t4)
        addi $t3, $t3, 4
        jal iterate_list

        move $s0, $zero
        addi $s0, $s0, 400

```

Question 3

This question's aim is recursively shuffling a string according to given number of operation. We have started our program with prompt user to enter inputs.

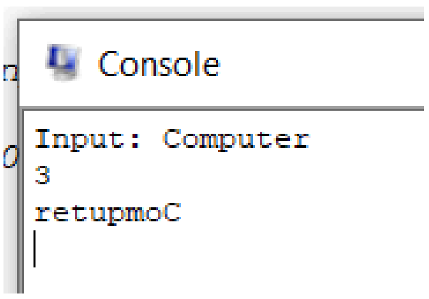
To shuffle the string we need to know that what is the length of the string. Because of that there is a **strlen** procedure. After calling this function, we found the first, last and middle indexes of the input we needed. Then we have invoked recursive function that name is **modify.**

This function rearranges the string according to a specific rule. First, the values of the registers at the point where the function is called are stored. New values are then assigned to the registers required for the function to run. The function continues or terminates the process by checking the second input value (which specifies how many times to shuffle). If processing continues, the string is split and each substring is passed back to the modify function. After each recursive call, the logger values stored in previous operations are restored. Finally, when the function is completed, it is returned to where the operation was called and any necessary cleanup is performed**.**

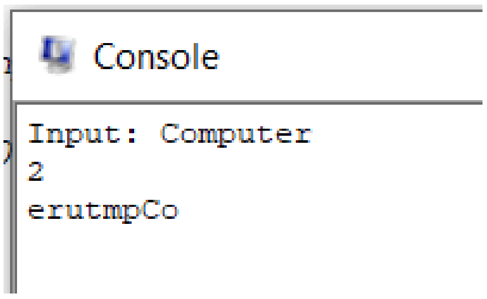

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text

```
PC       = 40009c
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = a
R3  [v1] = 7
R4  [a0] = 10010000
R5  [a1] = 0
R6  [a2] = 7
R7  [a3] = 2
R8  [t0] = 4
R9  [t1] = 10010001
R10 [t2] = 1
R11 [t3] = 72
R12 [t4] = 65
R13 [t5] = 10010000
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
[00400024] 34020004  ori $2, $0, 4              ; 7: li $v0, 4
[00400028] 3c011001  lui $1, 4097 [inputText] ; 8: la $a0, in
[0040002c] 34240080  ori $4, $1, 128 [inputText]
[00400030] 0000000c  syscall                    ; 9: syscall
[00400034] 34020008  ori $2, $0, 8              ; 12: li $v0, 8
[00400038] 3c011001  lui $1, 4097 [input]       ; 13: la $a0, i
[0040003c] 34240000  ori $4, $1, 0 [input]
[00400040] 34050014  ori $5, $0, 20             ; 14: li $a1, 2
[00400044] 0000000c  syscall                    ; 15: syscall
[00400048] 34020005  ori $2, $0, 5              ; 18: li $v0, 5
[0040004c] 0000000c  syscall                    ; 19: syscall
[00400050] 00023821  addu $7, $0, $2            ; 20: move $a3,
[00400054] 3c011001  lui $1, 4097 [input]       ; 22: la $t0, i
[00400058] 34280000  ori $8, $1, 0 [input]
[0040005c] 0c100032  jal 0x004000c8 [strlen]   ; 23: jal strle
[00400060] 3c011001  lui $1, 4097 [input]       ; 25: la $t5, i
[00400064] 342d0000  ori $13, $1, 0 [input]
[00400068] 3c011001  lui $1, 4097 [input]       ; 26: la $a0, i
[0040006c] 34240000  ori $4, $1, 0 [input]
[00400070] 34050000  ori $5, $0, 0              ; 27: li $a1, 0
[00400074] 00033021  addu $6, $0, $3            ; 28: move $a2,
[00400078] 20e7ffff  addi $7, $7, -1            ; 29: addi $a3,
[0040007c] 20c80001  addi $8, $6, 1             ; 31: addi $t0,
[00400080] 00084042  srl $8, $8, 1              ; 32: srl $t0,
[00400084] 0c100039  jal 0x004000e4 [modify]   ; 35: jal modif
[00400088] 34020004  ori $2, $0, 4              ; 36: li $v0, 4
[0040008c] 3c011001  lui $1, 4097 [input]       ; 37: la $a0, i
[00400090] 34240000  ori $4, $1, 0 [input]
[00400094] 0000000c  syscall                    ; 38: syscall
[00400098] 3402000a  ori $2, $0, 10             ; 40: li $v0, 1
[0040009c] 0000000c  syscall                    ; 41: syscall
```



```
Console
Input: Computer
3
retupmoC
```



```
Console
Input: Computer
2
erutmpCo
```

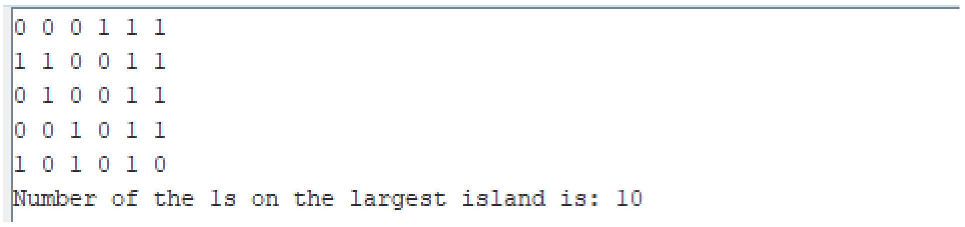
Question 4

In the 4th problem, main purpose is to determine the largest island of the matrix that is restored in the data segment. As it is seen in **Figure-3**, matrix is stored in the data segment.

```
.data
matrix: .byte 5, 6, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0
```

Figure-3: Matrix that is stored in the data segment

For the solution of the problem, the progam utilizes the Depth-First-Search approach. Algorithm advances until the end of the matrix. Then, the function rolls back recursively. The result of the execution of the program according to the matrix given in **Figure-3** is given in **Figure-4.**



```
0 0 0 1 1 1
1 1 0 0 1 1
0 1 0 0 1 1
0 0 1 0 1 1
1 0 1 0 1 0
Number of the 1s on the largest island is: 10
```

Figure-4: Output of the result