Report

Teacher, At first I write the function with recursive, but when I converting the recursive function to mips code on Mars, I got errors and couldn't fix it. Then I wrote it iterative and I know it will be given 15 points missing.

Also I didn’t bonus part.Tests worked correctly.

Function Descriptions

.data

```
.data
myArray:.space 100
screenMessage:.asciiz "Enter size of array:"
screenMessage2:.asciiz "Enter target number:"
screenMessage3:.asciiz "Enter elements of array:"
screenMessage4:.asciiz "Possible!\n"
screenMessage5: .asciiz "Not Possible!\n"
```

myArray keeps array elements in data.

And others variable keeps screen messages on data.

main:

```
jal addElementToArray
```

addElementToArray is is the loop that adds input taken from the user to the array.

In Mars

```
jal CheckSumPossibility #call CheckSumPossibility(num,arr,arraysize)

add  $t4,$zero,$v0 #return_val = CheckSumPossibility(num,arr,arraySize)
bne $t4,$zero,possible #if (return_val == 0)
```

In C++

```
int CheckSumPossibility(int num,int arr[], int size)
```

CheckSumPossibility: This function finds all the subset sums of the array, returns 1 if the total is equal to the given target number, otherwise it returns 0.

Possible: if the returned number is 1, "Possible!" prints.

CheckSumPossibility:

In Mars

```
first_loop:
beq $t2,$t1,first_loop_exit # i= 0 ; i<all_subset

addi $t3,$zero,0 #sums = 0
addi $t4,$zero,0 #j= 0
second_loop:
beq $t4,$a1,second_loop_exit # j= 0 ; j < size
```

In C++

```
for (int i=0; i<all_subset; i++){
    int sums = 0;
    for (int j=0; j<size; j++){
```

The first_loop and secon_loop procedures in Mars represent for loops in the C ++ code.

TESTS

1.Test

Array size :8 Target Num:129

Array Elements: 92 82 21 16 18 95 47 26

Output:Possible!

C++ Output

```
PS D:\Dersler\CSE 331 - Computer Organization\HW2> ./a.exe
8
129
92
82
21
16
18
95
47
26
Possible!
PS D:\Dersler\CSE 331 - Computer Organization\HW2>
```

Mars Output

```
Enter size of array:8
Enter target number:129
Enter elements of array:92
82
21
16
18
95
47
26
Possible!

-- program is finished running --
```

2.Test

Array size :8 Target Num:129

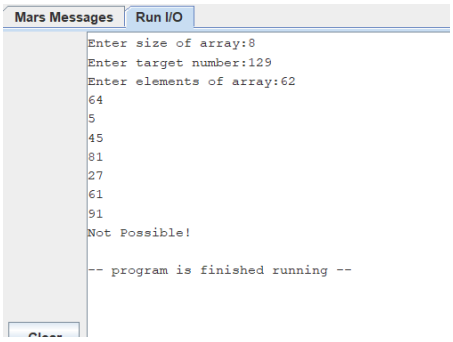
Array Elements: 62 64 5 45 81 27 61 91

Output:Not Possible!

C++ Output

```
PS D:\Dersler\CSE 331 - Computer Organization\HW2> ./a.exe
8
129
62
64
5
45
81
27
61
91
Not possible!
PS D:\Dersler\CSE 331 - Computer Organization\HW2>
```

Mars Output



3.Test

Array size :10 Target Num:42

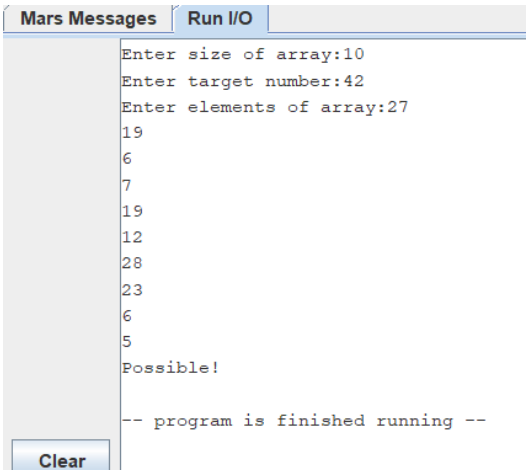
Array Elements: 27 19 6 7 19 12 28 23 6 5

Output:Possible!

C++ Output

```
PS D:\Dersler\CSE 331 - Computer Organization\HW2> ./a.exe
10
42
27
19
6
7
19
12
28
23
6
5
Possible!
PS D:\Dersler\CSE 331 - Computer Organization\HW2>
```

Mars Output



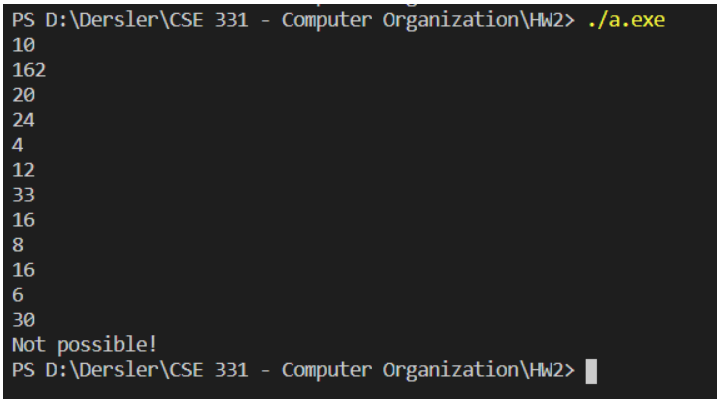
4.Test

Array size :10 Target Num:162

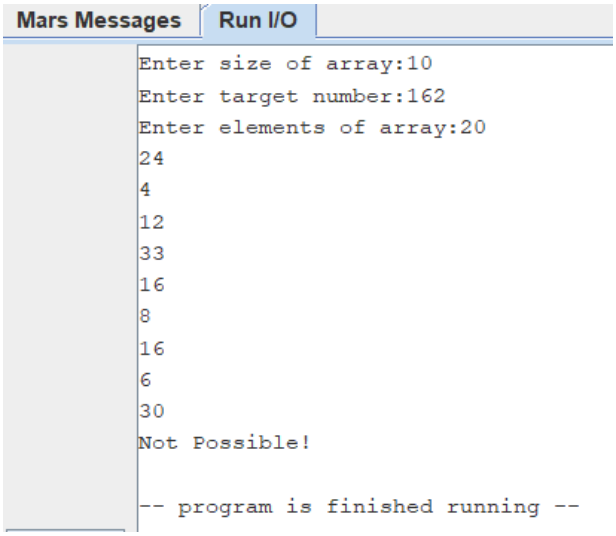
Array Elements: 20 24 4 12 33 16 8 16 6 30

Output:Not Possible!

C++ Output



Mars Output



5.Test

Array size :4 Target Num:32

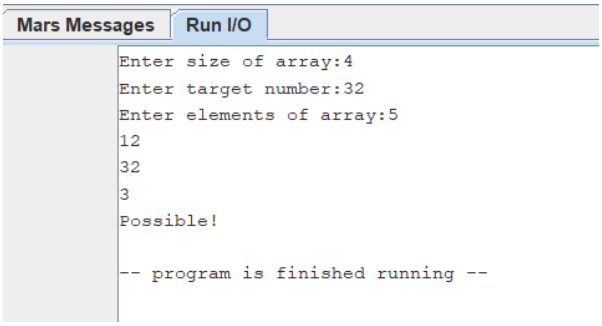
Array Elements: 5 12 32 3

Output:Possible!

C++ Output

```
PS D:\Dersler\CSE 331 - Computer Organization\HW2> ./a.exe
4
32
5
12
32
3
Possible!
PS D:\Dersler\CSE 331 - Computer Organization\HW2>
```

Mars Output



6.Test

Array size :5 Target Num:21

Array Elements: 7 8 2 3 1

Output: Possible!

C++ Output

```
PS D:\Dersler\CSE 331 - Computer Organization\HW2> ./a.exe
5
21
7
8
2
3
1
Possible!
PS D:\Dersler\CSE 331 - Computer Organization\HW2>
```

Mars Output