REPORT – ASSIGNMENT2

CAN DUYAR – 171044075

1)Readable explanation of functions in code(explanation of inputs and outputs, shortly explanation of function)

I used main function and a recursive function named as **"CheckSumPossibility"**.

Explanation of CheckSumPossibility function: I wrote it to find if a subset of array elements can sum up to the target num. If not possible then returns 0, otherwise returns 1. My CheckSumPossibility function works as recursively I have some conditions in this function and they check values of "num" and "arraySize" if num == 0 then it means that there is a possibility to obtain target value with sum of given numbers and in this case our recursive function returns 1.

If arraySize == 0 then it means that we didn't find a possibility to obtain target value so in this case our recursive function returns 0.

Explanation of main function: In this function, I got the "arraySize" and "num" values that I used as parameters from the user. and I asked the user to fill arr[ ]. I sent these values to my CheckSumPossibility function. In this case, CheckSumPossibility function returned an output as one or zero. If it returns as one then I printed it as "Possible" otherwise, "Not Possible"

My CheckSumPossibility function has 3 inputs:

Inputs:

num: This is our target number which is given as parameter. When I sent the num value which is given by user as parameter, I used $a1 register in my assembly code.

arr[ ]: This is our array that keeps user's integer inputs. When I sent arr[ ] that elements' are entered by user as parameter, I used $a2 register in my assembly code.

arraySize: This is number of inputs which is given as parameter. When I sent the arraySize value which is given by user as parameter, I used $a3 register in my assembly code.

Output:

This function returns 1 or 0. I used this return value in $v1 register in my assembly code

Explanation of labels:

rd_loop : it's same as for loop in my assembly code. I used it for getting array-inputs from the user in my main function.

P: I used this label to print "Possible!" in a condition.

NP: I used this label to print “Not possible!” in a condition.

done: This label tells the system that the program is done

label : I used it to control last return part. “return …. || ….. ” part works properly with the help of this label.

condition: It returns 1, if num == 0

condition2: It returns 0, if arraySize == 0

condition3: it returns CheckSumPossibility(num, arr, arraySize - 1), when (arr[arraySize - 1] > num) is true.

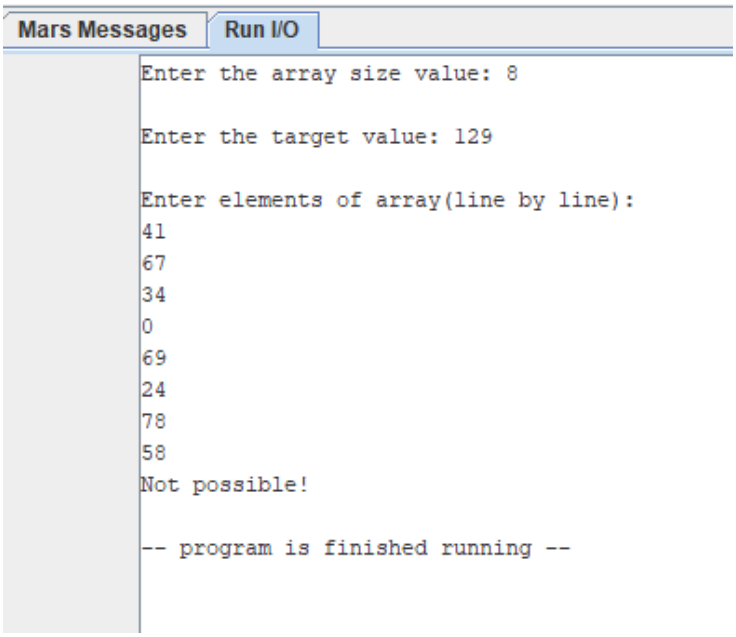
CheckSumPossibility_end: end of my function, to restores registers.

2) Result of test cases with screenshots & basic explanation.

I used same test scenarios as pdf. if you want you can also test it different test scenarios.

TEST-1 (num = 129 , arraySize = 8, arr: 41 67 34 0 69 24 78 58 ::::: Not possible!)

MIPS:



C++ :

```
can@can-VirtualBox:~/Desktop$ g++ hw2.cpp
can@can-VirtualBox:~/Desktop$ ./a.out
8 129
41 67 34 0 69 24 78 58
Not possible!
can@can-VirtualBox:~/Desktop$
```

TEST-2 (num = 129 , arraySize = 8, arr: 62 64 5 45 81 27 61 91 ::::: Not possible!)

MIPS:

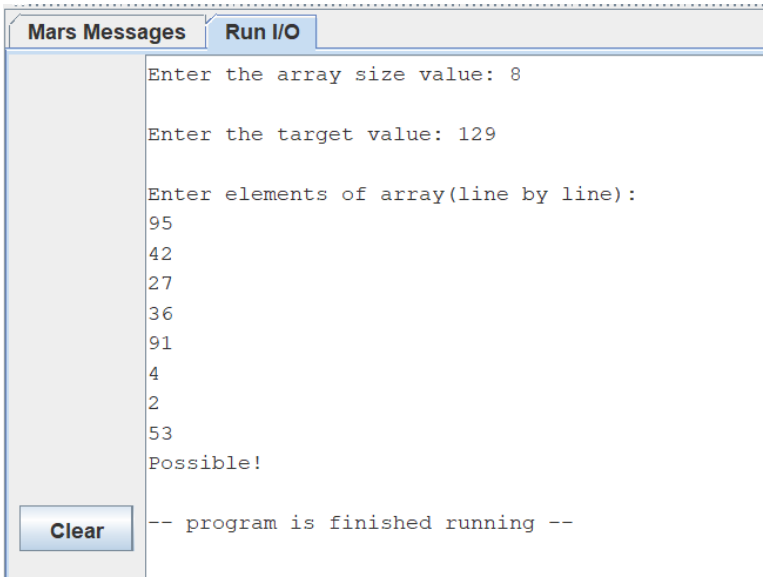
```
Mars Messages | Run I/O

Enter the array size value: 8

Enter the target value: 129

Enter elements of array(line by line):
62
64
5
45
81
27
61
91
Not possible!

-- program is finished running --
```

C++ :

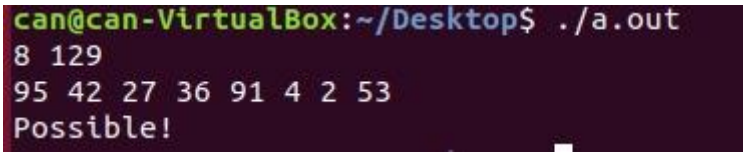
```
can@can-VirtualBox:~/Desktop$ g++ hw2.cpp
can@can-VirtualBox:~/Desktop$ ./a.out
8 129
62 64 5 45 81 27 61 91
Not possible!
can@can-VirtualBox:~/Desktop$
```

TEST-3 (num = 129 , arraySize = 8, arr: 95 42 27 36 91 4 2 53::::: Possible!)

MIPS:



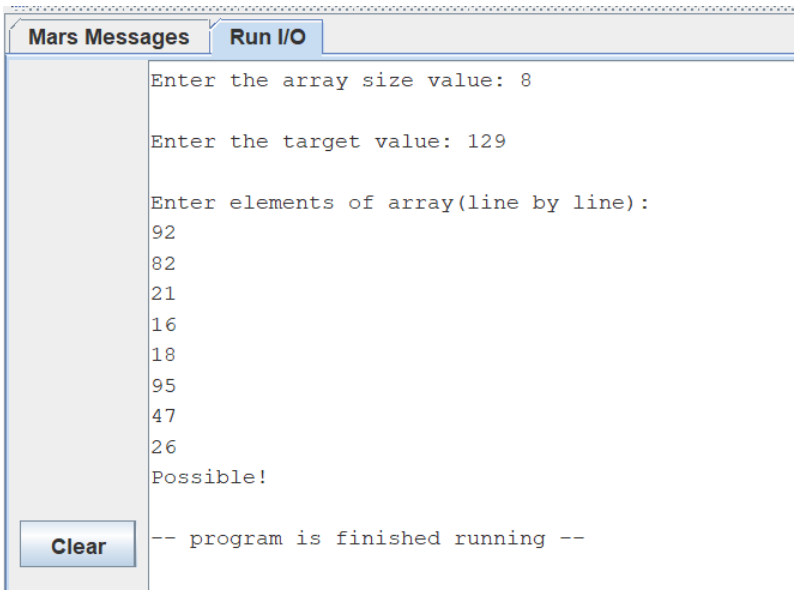
C++ :



36+91+2 = 129 so it's possible!

TEST-4 (num = 129 , arraySize = 8, arr: 92 82 21 16 18 95 47 26 :::: Possible!)

MIPS:



C++ :

```
can@can-VirtualBox:~/Desktop$ g++ hw2.cpp
can@can-VirtualBox:~/Desktop$ ./a.out
8 129
92 82 21 16 18 95 47 26
Possible!
```

92 + 21 + 16 = 129 so it's possible!

TEST-5 (num = 129 , arraySize = 8, arr: 71 38 69 12 67 99 35 94 :::: Possible!)

MIPS:

```
Mars Messages   Run I/O

Enter the array size value: 8

Enter the target value: 129

Enter elements of array(line by line):
71
38
69
12
67
99
35
94
Possible!

-- program is finished running --

Clear
```

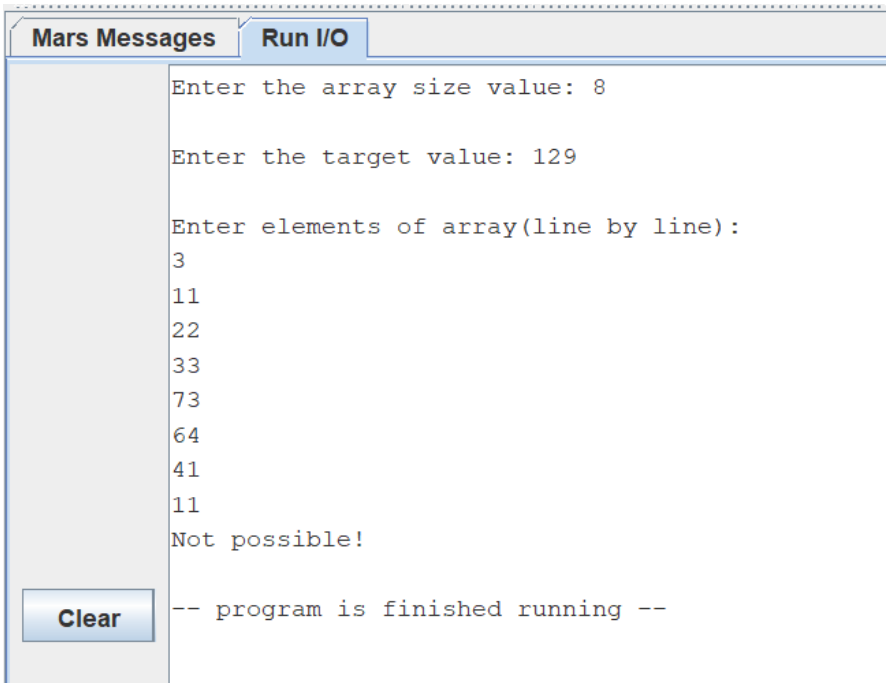
C++ :

```
can@can-VirtualBox:~/Desktop$ g++ hw2.cpp
can@can-VirtualBox:~/Desktop$ ./a.out
8 129
71 38 69 12 67 99 35 94
Possible!
```

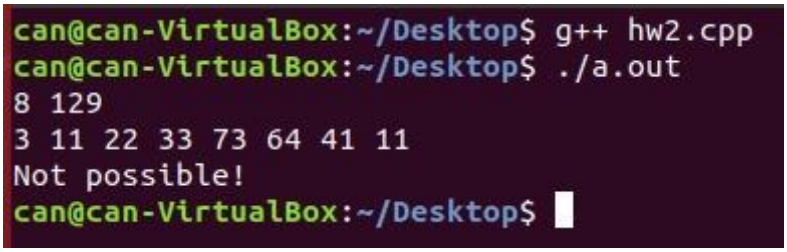
35 + 94 = 129 so it's possible!

TEST-6 (num = 129 , arraySize = 8, arr: 3 11 22 33 73 64 41 11 :::: Not possible!)

MIPS:



C++ :



3) Explain if you have a missing parts, bonus parts or adding parts.

-> Everything is done, I also test my code with different scenarios except pdf.

-> I also tried to do the bonus part but it didn't work properly, I added some comment lines about bonus part in my cpp and assembly codes,please also check all of them. It does not work properly but i tried to write something. I would be very happy,if I can can get some points from the bonus part 😊