Homework 2 Report

Elif Akgün
1801042251

1. Design Explanation

In this program the task is to find if a subset of array elements can sum up to the target num. If not possible the output will be “Not possible!”. If it is possible, output is “Possible!”. Every array element can use only once.and only positive integers are allowed as array elements. Finding only one combination is enough to output “Possible!”.
As solution, I used the backtracking technique.The backtracking technique generates every subset only once. There are two options for every element in the set, either subset includes this element or it does not. I set the maximum size of the array to 100. So the size of a subset can be maximum 100. That’s why, I created the same size *subset* array to store subset elements. There is a flag to notify target sum is got. Also there is an variable *k* which keeps current size of subset array and its current index. In for loop, kth index of subset array becomes array’s ith element. Then with recursive call, it moves to next element of array. After recursive call, it checks sum of subset array’s elements. *sum* variable keeps sum of the each subset’s elements. If sum equals target sum, then flag becomes 1 to notify expected sum got. If sum does not equal to target sum, k decreases 1 for backtracking and it goes on like this. When target sum is found, then recursive calls don’t enter for loop and return. Finally, when target sum is found, *CheckSumPossibility* function returns 1, otherwise it returns 0.
Also I did bonus part. I kept related subset’s elements in the subset array. Then I printed it to console after ”Subset: ” message.

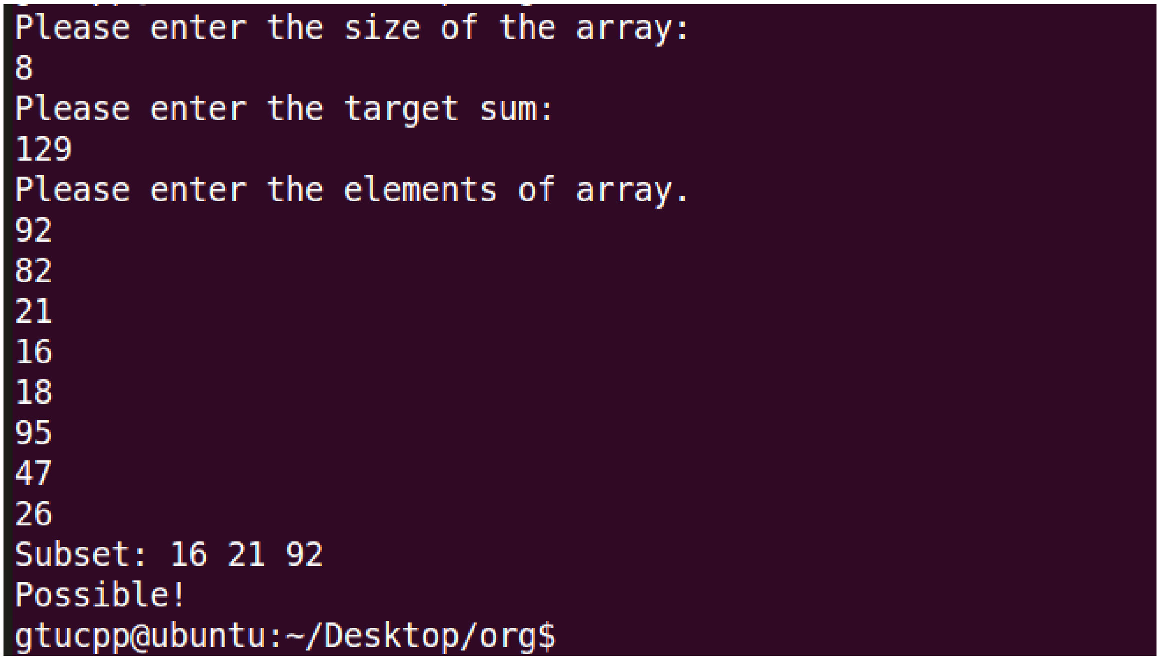
2. Usage

In both parts, to run program, you should enter array’s size after ”Please enter the size of the array:” message. Then, you should enter the target sum after ”Please enter the target sum:” message. Finally, you should enter the array elements after ”Please enter the elements of array:” message. You can enter as many elements as the size of array. After enter elements, result shoul be come to console. I set the maximum size of the array to 100 for both part. You should not enter a number bigger than 100 for size.

3. Tests

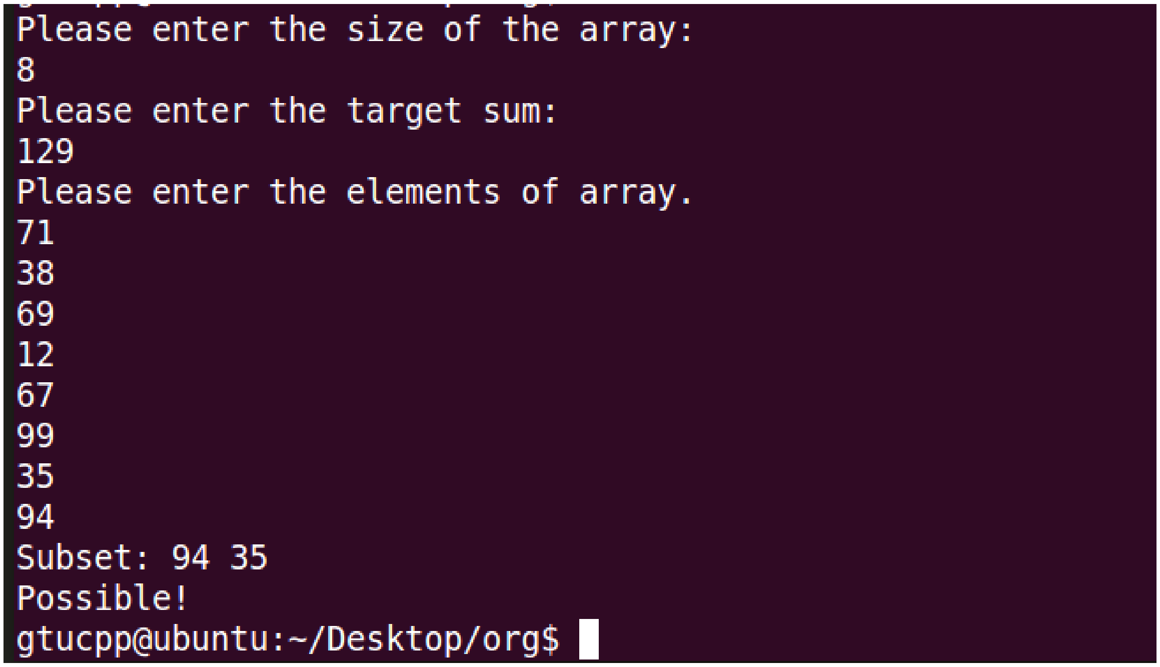
3.1 Part 1

Figure 1: C++ Test 1



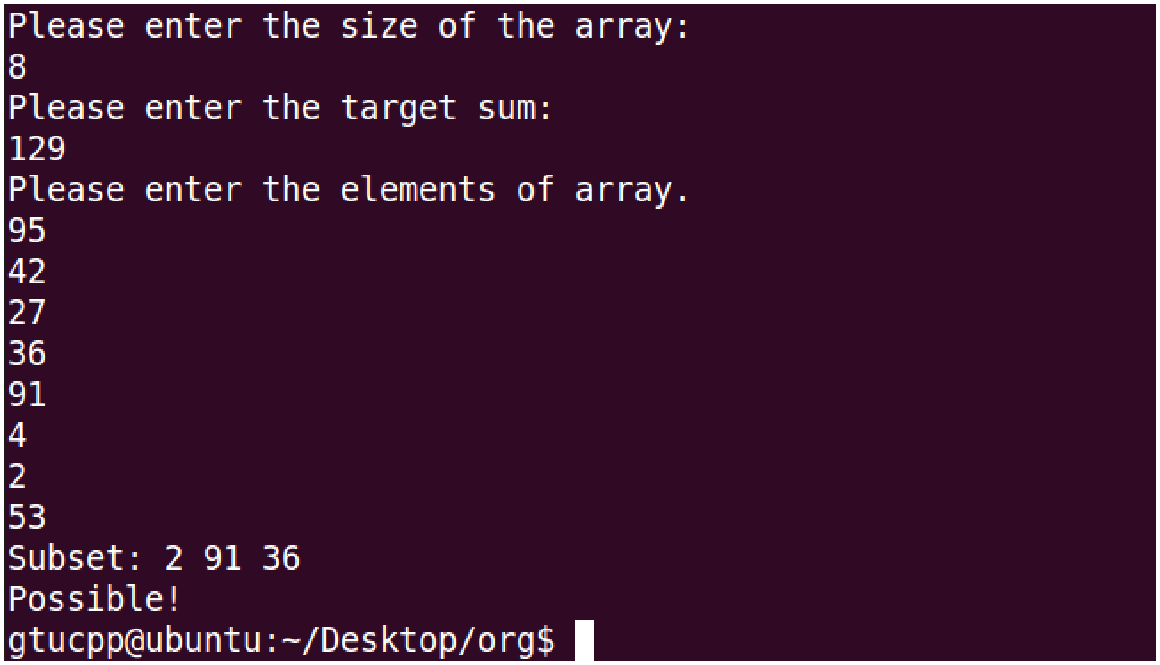
```
Please enter the size of the array:
8
Please enter the target sum:
129
Please enter the elements of array.
92
82
21
16
18
95
47
26
Subset: 16 21 92
Possible!
gtucpp@ubuntu:~/Desktop/org$
```

Figure 2: C++ Test 2



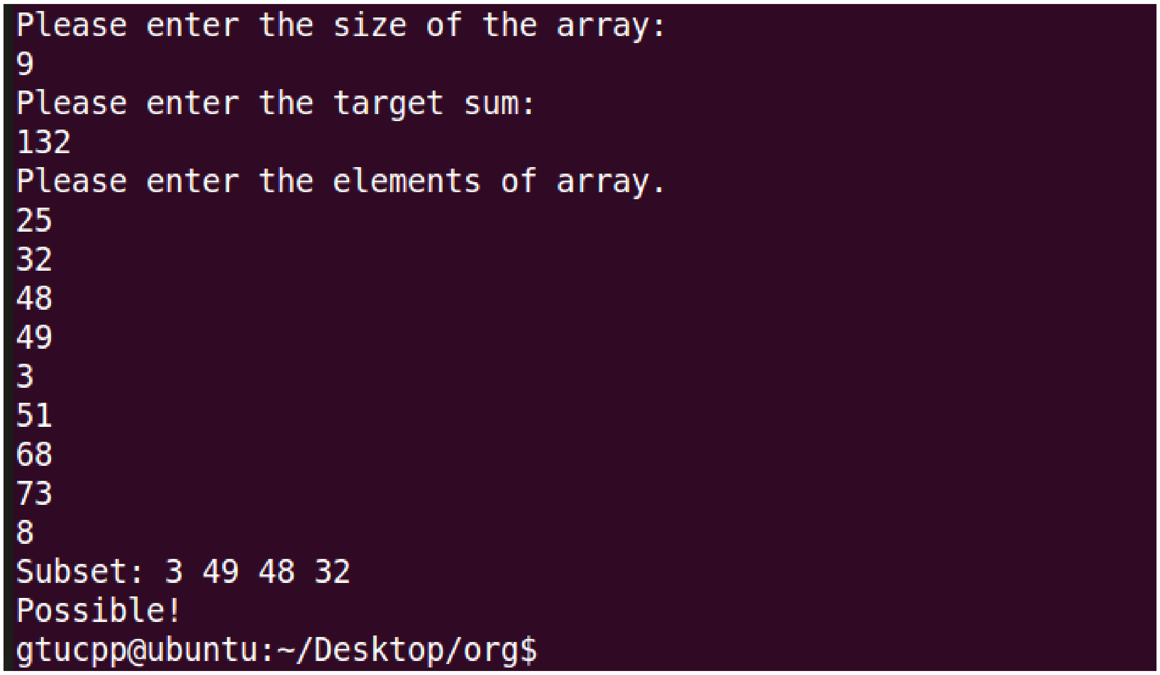
```
Please enter the size of the array:
8
Please enter the target sum:
129
Please enter the elements of array.
71
38
69
12
67
99
35
94
Subset: 94 35
Possible!
gtucpp@ubuntu:~/Desktop/org$
```

Figure 3: C++ Test 3



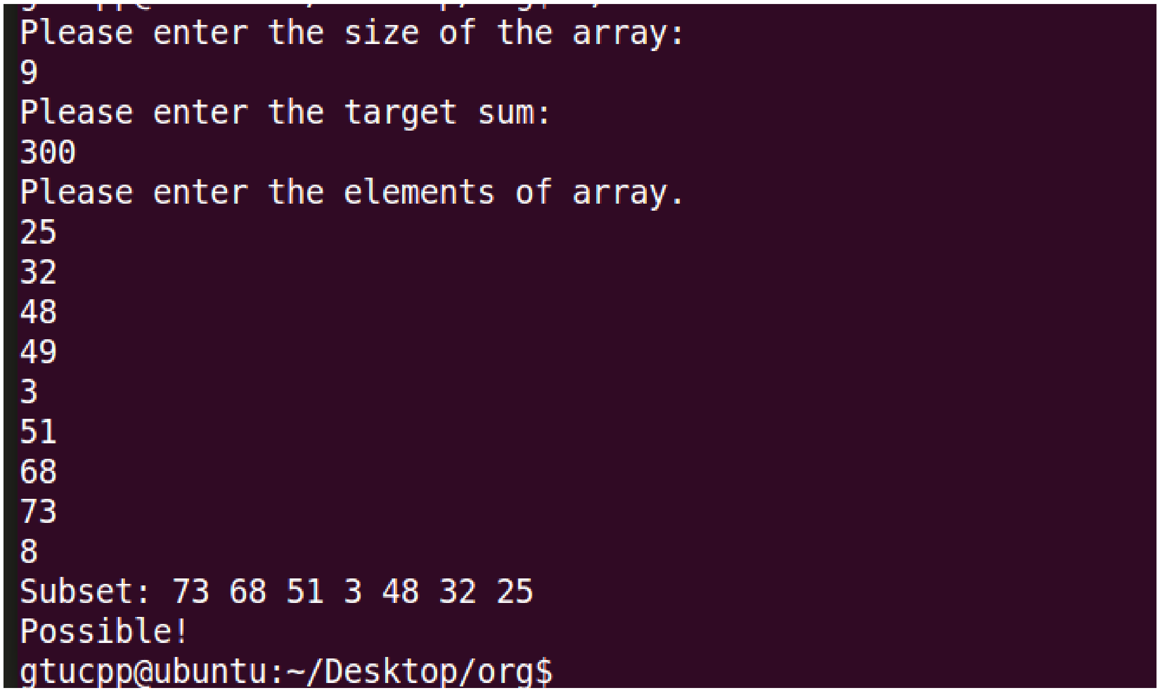
```
Please enter the size of the array:
8
Please enter the target sum:
129
Please enter the elements of array.
95
42
27
36
91
4
2
53
Subset: 2 91 36
Possible!
gtucpp@ubuntu:~/Desktop/org$
```

Figure 4: C++ Test 4



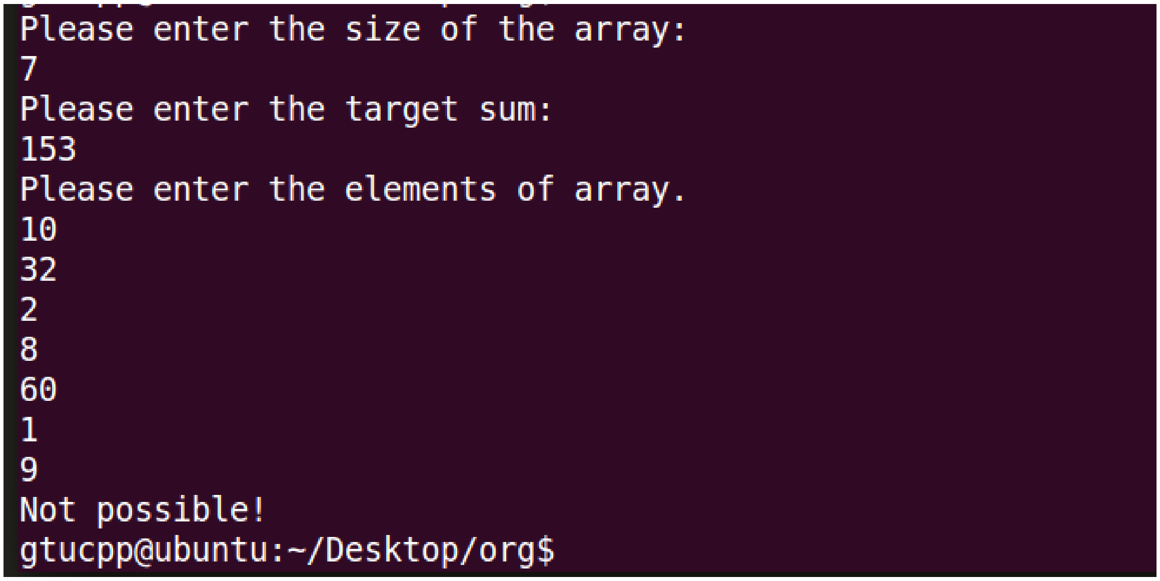
```
Please enter the size of the array:
9
Please enter the target sum:
132
Please enter the elements of array.
25
32
48
49
3
51
68
73
8
Subset: 3 49 48 32
Possible!
gtucpp@ubuntu:~/Desktop/org$
```

Figure 5: C++ Test 5



```
Please enter the size of the array:
9
Please enter the target sum:
300
Please enter the elements of array.
25
32
48
49
3
51
68
73
8
Subset: 73 68 51 3 48 32 25
Possible!
gtucpp@ubuntu:~/Desktop/org$
```

Figure 6: C++ Test 6



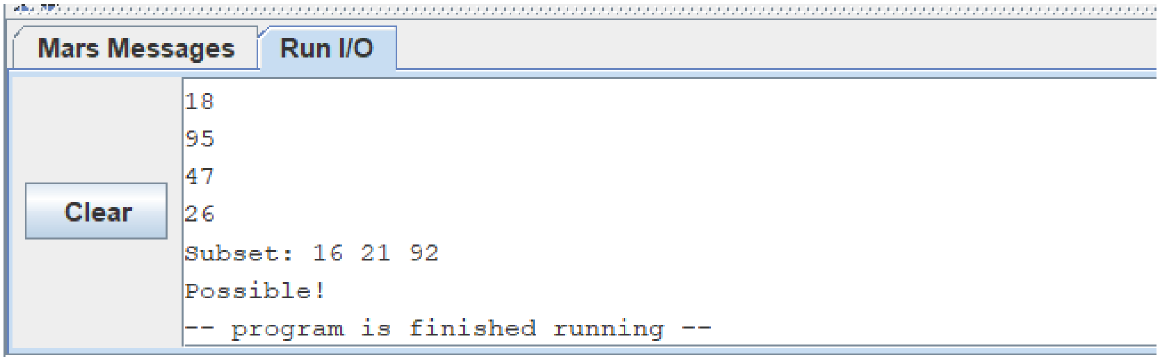
```
Please enter the size of the array:
7
Please enter the target sum:
153
Please enter the elements of array.
10
32
2
8
60
1
9
Not possible!
gtucpp@ubuntu:~/Desktop/org$
```

3.2 Part 2

MIPS Assembly Test 1:
Please enter the size of the array: 8
Please enter the target sum: 129

Please enter the elements of array:
92 82 21 16 18 95 47 26

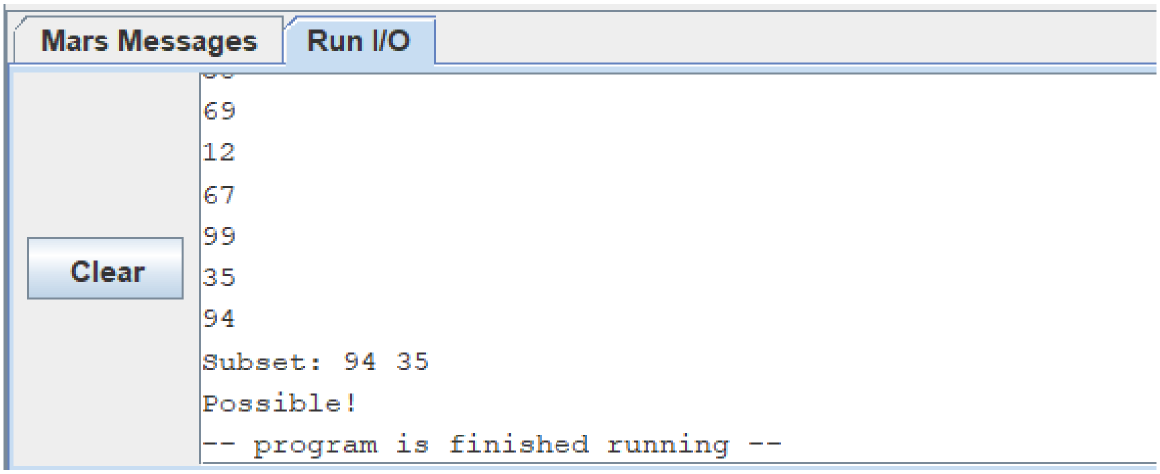
Figure 7: MIPS Assembly Test 1



```
18
95
47
26
Subset: 16 21 92
Possible!
-- program is finished running --
```

MIPS Assembly Test 2:
Please enter the size of the array: 8
Please enter the target sum: 129
Please enter the elements of array:
71 38 69 12 67 99 35 94

Figure 8: MIPS Assembly Test 2

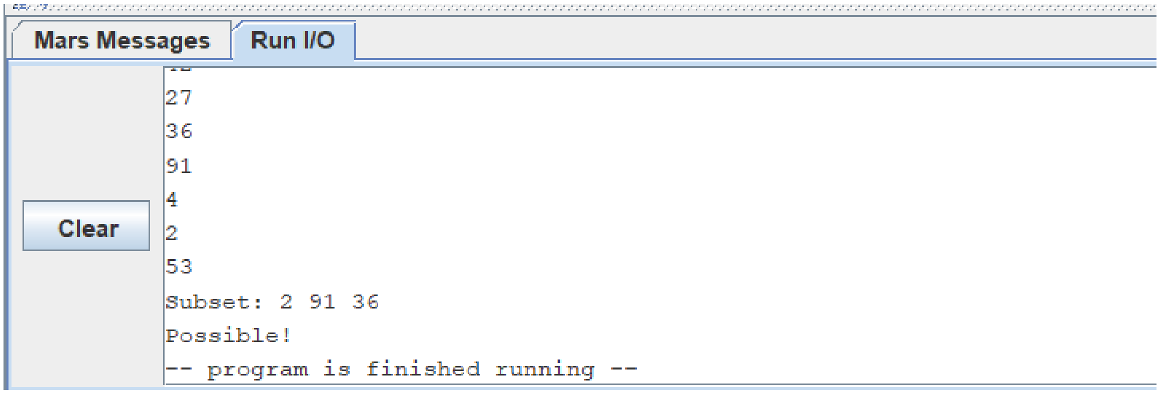


```
69
12
67
99
35
94
Subset: 94 35
Possible!
-- program is finished running --
```

MIPS Assembly Test 3:

Please enter the size of the array: 8
Please enter the target sum: 129
Please enter the elements of array:
95 42 27 36 91 4 2 53

Figure 9: MIPS Assembly Test 3

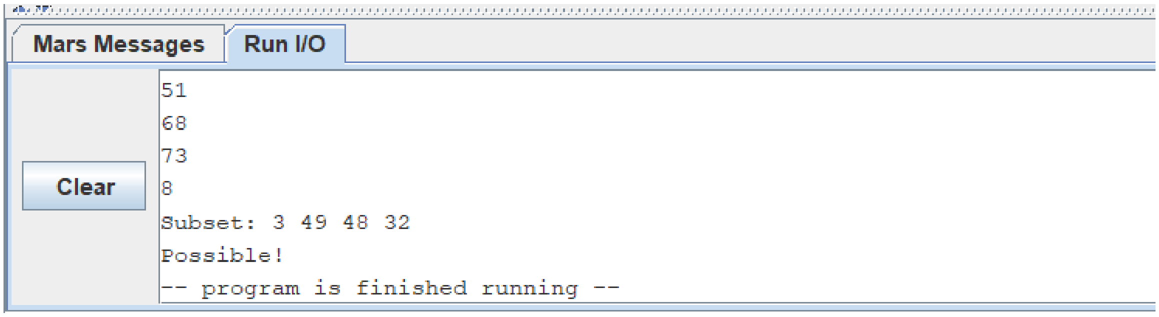


```
27
36
91
4
2
53
Subset: 2 91 36
Possible!
-- program is finished running --
```

MIPS Assembly Test 4:

Please enter the size of the array: 9
Please enter the target sum: 132
Please enter the elements of array:
25 32 48 49 3 51 68 73 8

Figure 10: MIPS Assembly Test 4

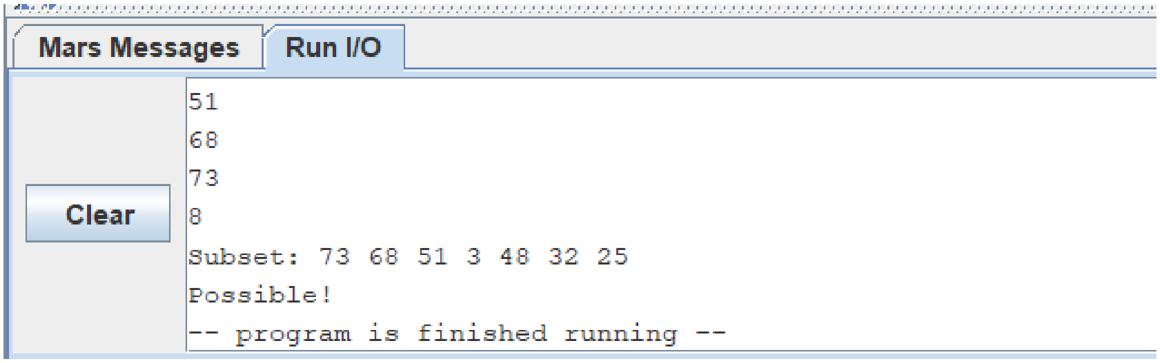


```
51
68
73
8
Subset: 3 49 48 32
Possible!
-- program is finished running --
```

MIPS Assembly Test 5:

Please enter the size of the array: 9
Please enter the target sum: 300
Please enter the elements of array:
25 32 48 49 3 51 68 73 8

Figure 11: MIPS Assembly Test 5

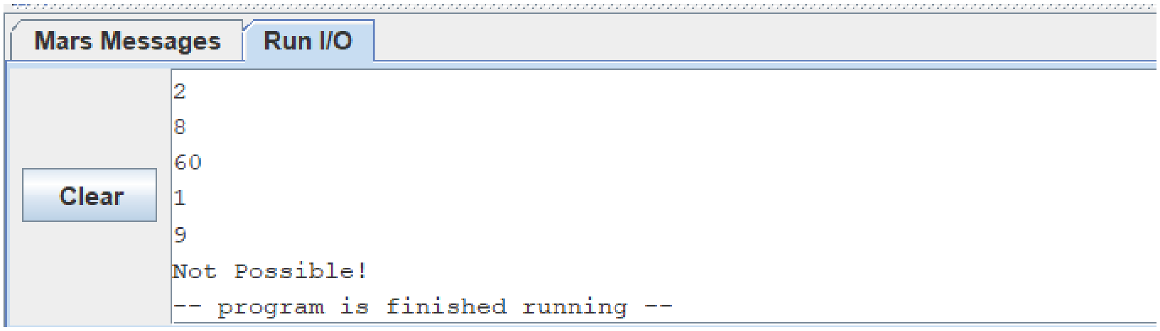


```
51
68
73
8
Subset: 73 68 51 3 48 32 25
Possible!
-- program is finished running --
```

MIPS Assembly Test 6:

Please enter the size of the array: 7
Please enter the target sum: 153
Please enter the elements of array:
10 32 2 8 60 1 9

Figure 12: MIPS Assembly Test 6



```
2
8
60
1
9
Not Possible!
-- program is finished running --
```