CSE 331/503 COMPUTER ORGANIZATION

HOMEWORK 2

OZAN GEÇKİN

1801042103

Proje:

You will be given an integer array arr and a number num. The task is to find if a subset of array elements can sum up to the target num. If not possible you will output “Not possible!”. If it is possible, output “Possible!”. You can use every array element only once. Only positive integers are allowed as array elements. Finding only one combination is enough to output “Possible!”.

Algorithm:

I get three parameters from the main function. These are array, array size, target sum number. I wrote a recursive function that returns integer. I described my algorithm as a tree structure. Each node is divided into two. Nodes include primarily size and target sum number. On the left side of the node I decrease size 1 and continue as the target number, on the right, I continue by decreasing size 1 and removing the elements of the array from the target number. Then I return 1 or 0 by checking the size and num in my base cases.

Report to hw2.cpp:

Function:

1)Main function:

I get the parameters I will use in my recursive function from the user and I write possible or not possible on the screen according to the result of the recursive function.

2) CheckSumPossibility:

Recursive function There are 2 base cases, one is the target number equality of zero and the other is your array size being equal to zero or less than zero and target sum number less than zero checking. My other control is checking the target number with the last element of the array. If the target number is less than the last element of the array, I remove the last element and call the function again. If the last element is not less than the target number. I call recursive function twice. In one I am sending the last element the the recursive fuction. In the other, I remove the last element from the target number and send the target number that I just created and a decrease to you. Then I compare these two recursive results with the or logic operator and return.

Test:

```
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 10
Please Enter Target Sum Number: 12
Please Enter array element 1: 30
Please Enter array element 2: 30
Please Enter array element 3: 17
Please Enter array element 4: 31
Please Enter array element 5: 29
Please Enter array element 6: 30
Please Enter array element 7: 26
Please Enter array element 8: 30
Please Enter array element 9: 10
Please Enter array element 10: 25
Not possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 3
Please Enter Target Sum Number: 6
Please Enter array element 1: 1
Please Enter array element 2: 2
Please Enter array element 3: 3
Possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 10
Please Enter Target Sum Number: 12
Please Enter array element 1: 9
Please Enter array element 2: 21
Please Enter array element 3: 32
Please Enter array element 4: 2
Please Enter array element 5: 30
Please Enter array element 6: 17
Please Enter array element 7: 25
Please Enter array element 8: 22
Please Enter array element 9: 2
Please Enter array element 10: 12
Possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 10
Please Enter Target Sum Number: 10000
Please Enter array element 1: 9
Please Enter array element 2: 21
Please Enter array element 3: 32
Please Enter array element 4: 2
Please Enter array element 5: 30
Please Enter array element 6: 17
Please Enter array element 7: 25
Please Enter array element 8: 22
Please Enter array element 9: 2
Please Enter array element 10: 12
Not possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$
```

```
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 10
Please Enter Target Sum Number: 62
Please Enter array element 1: 8
Please Enter array element 2: 33
Please Enter array element 3: 15
Please Enter array element 4: 28
Please Enter array element 5: 25
Please Enter array element 6: 11
Please Enter array element 7: 32
Please Enter array element 8: 33
Please Enter array element 9: 26
Please Enter array element 10: 11
Possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$
```

```
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ g++ hw2.cpp -o hw2
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 8
Please Enter Target Sum Number: 129
Please Enter array element 1: 41
Please Enter array element 2: 67
Please Enter array element 3: 34
Please Enter array element 4: 0
Please Enter array element 5: 69
Please Enter array element 6: 24
Please Enter array element 7: 78
Please Enter array element 8: 58
Not possible!
ozan@ozan-C650-NOTEBOOK-DISCRETE:~/Desktop/organizasyon$ ./hw2
Please Enter Array Size: 8
Please Enter Target Sum Number: 129
Please Enter array element 1: 95
Please Enter array element 2: 42
Please Enter array element 3: 27
Please Enter array element 4: 36
Please Enter array element 5: 91
Please Enter array element 6: 4
Please Enter array element 7: 2
Please Enter array element 8: 53
Possible!
```

Report to hw2.asm:

Before writing my assembly code, I will use strings ,array ,size and target sum number ,I reseved places in the data for sum. I received my inputs from the user in main. While get, I filled in my registers by adhering to the contract with the registers. As in C++ code, there are 2 functions in assembly code, main and CheckSumPossibility. When the descriptions of the functions are the same as in the cpp code, I did not rewrite it so that the report would not be extended.

Inputs:

size: I take it from the user with syscall for array size and put it in the $a1 register ,which is the parameter register.

sum: I take from the user with syscall for target sum number and put it in the $a2 register, which is the parameter register.

arr: It holds integers from user and put it in the $a3 register which is the parameter register.

Output:

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

Explain label:

inputArray: . I used it for getting array-inputs from the user in my main function.

exitInputArray: I used it for exit inputArray safely.

CheckSumPossiliytArgumantControl: It label reduce size one and call again function.

CheckSumPossiliytArgumantControl2: It label reduce size one and subtract the array element from the target sum and call again function.

Return1:if CheckSumPossiliyt is true ,assign $v1,1.

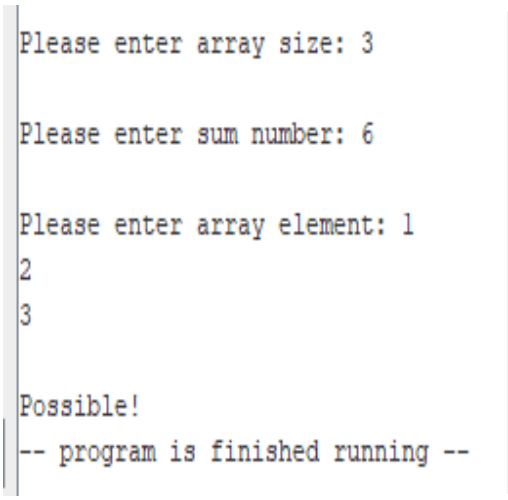
Return0: if CheckSumPossiliyt is false ,assign $v1,0.

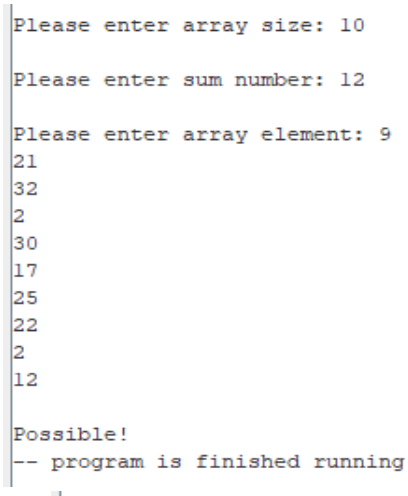
exitCheckSumPossibility: It label end of function and restore register.

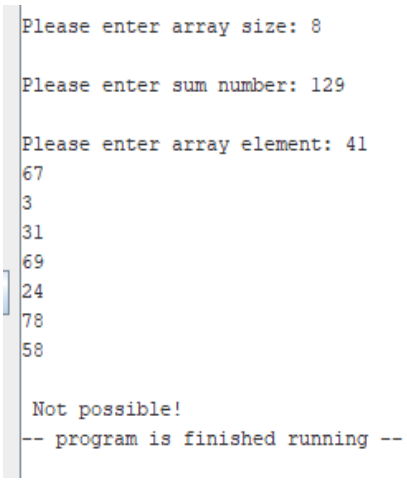
possibility: It labels print possible and exit.

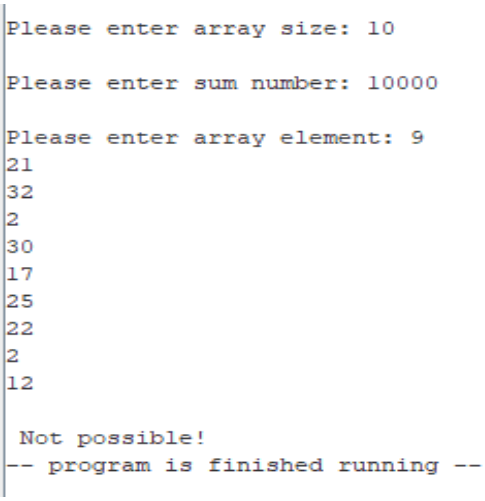
notPossibility: It labels print not possible and exit.

Test:









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

In my cpp code everythin works exactly. I have done many tests and all are success.

One thing does not work in the assembly code. I have been study for this for days. I tried to do it until the deadline, but it didn't. I can't do is call two recursive function on the same line and can't use logic operator.(return CheckSumPossibility(num, arr, size-1)||CheckSumPossibility(num-arr[size-1], arr,size-1);) .I am calling the functions separately here.