

Q4,

Pseudo-code:

Step 1: check if there is a subsequence of the given sequence with sum equal to given sum:

```

def isSubSum(st, n, sm):
    # The value of subset[i][j] will be
    # true if there is a subset of
    # set[0..j-1] with sum equal to i

    boolean[][] solution = new boolean[A.length + 1][sum + 1]

    # If sum is 0, then answer is true
    for i in range(0, n + 1):
        subset[i][0] = True

    # If sum is not 0 and set is empty,
    # then answer is false
    for i in range(1, sm + 1):
        subset[0][i] = False

    # Fill the subset table in bottom
    # up manner
    for i in range(1, n + 1):
        for j in range(1, sm + 1):
            if (j < st[i - 1]):
                subset[i][j] = subset[i - 1][j]
            if (j >= st[i - 1] and subset[i][j] == False):
                subset[i][j] = subset[i - 1][j] or subset[i - 1][j - st[i - 1]]

    return subset[n][sm];

```

		Sum						
Elements		0	1	2	3	4	5	6
	0	T	F	F	F	F	F	F
	3	T	F	F	T	F	F	F
	2	T	F	T	T	F	T	F
	7	T	F	T	T	F	T	F
	1	T	T	T	T	T	T	T

Step 2: if subset == true, return the sequence; else, return false
 So I just include the current element whenever I move left:

		Sum						
		0	1	2	3	4	5	6
Elements	0	T	F	F	F	F	F	F
	3	T	F	F	T	F	F	F
	2	T	F	T	T	F	T	F
	7	T	F	T	T	F	T	F
	1	T	T	T	T	T	T	T

Include the current element whenever you move left.

Final code:

```

1  # Returns true if there is a subset
2  # of set[] with sum equal to given sum
3  def isSubsetSum(st, n, sm):
4      # The value of subset[i][j] will be
5      # true if there is a subset of
6      # set[0..j-1] with sum equal to i
7      subset = [[False for i in range(sm+1)] for y in range(n+1)]
8
9      # If sum is 0, then answer is true
10     for i in range(0, n + 1):
11         subset[i][0] = True
12
13     # If sum is not 0 and set is empty,
14     # then answer is false
15     for i in range(1, sm + 1):
16         subset[0][i] = False
17
18     # Fill the subset table in bottom
19     # up manner
20     for i in range(1, n + 1):
21         for j in range(1, sm + 1):
22             if (j < st[i - 1]):
23                 subset[i][j] = subset[i - 1][j]
24             if (j >= st[i - 1]):
25                 subset[i][j] = subset[i - 1][j] or subset[i - 1][j - st[i - 1]]
26
27     print(subset[n][sm])
28     FindSequence(subset, st, n, sum)
29
30
31
32

```

```

33 def FindSequence(subset,st,n,sum):
34     answer = []
35     row = n
36     col = sum
37     if subset[row][col] == False:
38         return False
39     else:
40         while row>=1 and col >=1:
41             if subset[row][col] == True:
42                 if subset[row-1][col] == False:
43                     col = col -1
44                     if st[row-1] not in answer:
45                         answer.append(st[row-1])
46                         sum = sum -st[row-1]
47                         if (sum == 0):
48                             break
49                     continue
50                 else:
51                     row = row -1
52                     continue
53             else:
54                 col = col -1
55         #for i in range(0,len(answer)):
56         print(answer)
57         #print(subset)
58         return 0
59
60
61 # Driver program to test above function
62 set = [2,7,1,3]
63 sum = 6
64 n = len(set)
65 print(isSubsetSum(set, n, sum))
66
67

```

output:

```

59
60
61 # Driver program to test above function
62 set = [3,2,7,1]
63 sum = 6
64 n = len(set)
65 print(isSubsetSum(set, n, sum))
66
67
68 #subset=[[0] * (sm+1)] * (n+1)
69 #subset[2][1] = True
70 #print(subset)

```

Run subsetsum

```

/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/wushuo/PycharmProjects/cse541/subsetsum.py
True
[1, 2, 3]
None
Process finished with exit code 0

```

input: set = [3,2,7,1] sum =6

output answer = [1,2,3]

worst case: $T(n) = \Theta(\text{sum} * n) + \Theta(n + \text{sum}) = \Theta(\text{sum} * n)$ since we need to fill the table with size $\text{sum} * n$.

where n is the length of the sequence. Sum is the number we want to find.