WEEK 15

Pointers

Roll no : 240701095

Name : Darshini.H

Attempt 1

Status	Finished
Started	Monday, 13 January 2025, 6:55 PM
Completed	Monday, 13 January 2025, 7:01 PM
Duration	6 mins 11 secs

PROBLEM 1 : Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

Example
arr = [1, 3, 2, 4, 5]
Return the array [5, 4, 2, 3, 1] which is the reverse of the input array.

Function Description
        Complete the function reverseArray in the editor below.

reverseArray has the following
        parameter(s): int arr[n]: an array of
        integers

Return
        int[n]: the array in reverse order

Constraints
        $1 \leq n \leq 100$
        $0 < arr[i] \leq 100$

Input Format For Custom Testing
The first line contains an integer, n, the number of elements in arr.
Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer, arr[i].

Sample Input
1
3
2
4
5

Sample Output
4
2
3
1

CODE:

```
35   int* reverseArray(int arr_count, int *arr, int *result_count)
36 ▼ {
37       *result_count=arr_count;
38       int* result=(int*)malloc(arr_count* sizeof(int));
39       for(int i=0;i<arr_count;i++)
40       result[i]=arr[arr_count-1-i];
41       return result;
42
43   }
44
```

OUTPUT:

	Test	Expected	Got	

✓	`int arr[] = {1, 3, 2, 4, 5};`
	`int result_count;`
	`int* result = reverseArray(5, arr, &result_count);`
	`for (int i = 0; i < result_count; i++)`
	`        printf("%d\n", *(result + i));`
		5
		4
		2
		3
		1
			5
			4
			2
			3
			1
				✓

Passed all tests! ✓

PROBLEM 2 : An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of minLength or more, and it can only make one cut at a time. Given the array lengths [] representing the desired lengths of each segment, determine if it is possible to make the necessary cuts using this machine. The rod is marked into lengths already, in the order given.

Examp
le n = 3
lengths = [4, 3, 2]
minLength = 7

The rod is initially sum(lengths) = 4 + 3 + 2 = 9 units long. First cut off the segment of length 4 + 3 = 7 leaving a rod 9 - 7 = 2. Then check that the length 7 rod can be cut into segments of lengths 4 and 3. Since 7 is greater than or equal to minLength = 7, the final cut can be made. Return "Possible".

Examp
le n = 3
lengths = [4, 2, 3]
minLength = 7

The rod is initially sum(lengths) = 4 + 2 + 3 = 9 units long. In this case, the initial cut can be of length 4 or 4 + 2 = 6. Regardless of the length of the first cut, the remaining piece will be shorter than minLength. Because n - 1 = 2 cuts cannot be made, the answer is "Impossible".

Function Description
        Complete the function cutThemAll in the editor below.

cutThemAll has the following parameter(s):
        int lengths[n]: the lengths of the segments, in order
        int minLength: the minimum length the machine can accept

Returns
        string: "Possible" if all n-1 cuts can be made. Otherwise, return the string "Impossible".

Constraints
  - $2 \le n \le 105$
  - $1 \le t \le 109$
  - $1 \le lengths[i] \le 109$

The sum of the elements of lengths equals the uncut rod length.

INPUT FORMAT : The first line contains an integer, n, the number of elements in lengths.

Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer, lengths[i]. The next line contains an integer, minLength, the minimum length accepted by the machine.

Sample Input For Custom Testing

STDIN		Function
N		--------
4	→	lengths[] size n = 4
3	→	lengths[] = [3, 5, 4, 3]
5		
4		
3		
9	→	minLength= 9

Sample Output
Possible

Explanation

The uncut rod is 3 + 5 + 4 + 3 = 15 units long. Cut the rod into lengths of 3 + 5 + 4
= 12 and 3.
Then cut the 12-unit piece into lengths 3 and 5 + 4 = 9.
The remaining segment is 5 + 4 = 9 units and that is long enough to make the final cut.

CODE :

```c
29  char* cutThemAll(int lengths_count, long *lengths, long minLength)
30  {
31      int s=0;
32      for(int i=0;i<lengths_count-1;i++)
33      {
34          s=s+*(lengths+i);
35      }
36      if(s>=minLength)
37      {
38          return "Possible";
39      }
40      else
41      {
42          return "Impossible";
43      }
44
45  }
46
```

OUTPUT :

	Test	Expected	Got	
✓	long lengths[] = {3, 5, 4, 3}; printf("%s", cutThemAll(4, lengths, 9))	Possible	Possible	✓
✓	long lengths[] = {5, 6, 2}; printf("%s", cutThemAll(3, lengths, 12))	Impossible	Impossible	✓

Passed all tests! ✓