

```
1  /*
2   * Complete the 'reverseArray' function below.
3   *
4   * The function is expected to return an INTEGER_ARRAY.
5   * The function accepts INTEGER_ARRAY arr as parameter.
6   */
7
8  /*
9   * To return the integer array from the function, you should:
10   *   - Store the size of the array to be returned in the result_count
11   *   - Allocate the array statically or dynamically
12   *
13   * For example,
14   * int* return_integer_array_using_static_allocation(int* result_count)
15   *   *result_count = 5;
16   *
17   *   static int a[5] = {1, 2, 3, 4, 5};
18   *
19   *   return a;
20   * }
21   *
```

```
16 *  
17 *     static int a[5] = {1, 2, 3, 4, 5};  
18 *  
19 *     return a;  
20 * }  
21 *
```

```
18 *  
19 *     return a;  
20 * }  
21 *  
22 * int* return_integer_array_using_dynamic_allocation(int* result_co  
23 *     *result_count = 5;  
24 *  
25 *     int *a = malloc(5 * sizeof(int));  
26 *  
27 *     for (int i = 0; i < 5; i++) {  
28 *         *(a + i) = i + 1;  
29 *     }  
30 *  
31 *     return a;  
32 * }  
33 *
```

```

30  *
31  *    return a;
32  * }
33  *
34  */
35  int* reverseArray(int arr_count, int *arr, int *result_count) {
36      *result_count=arr_count;
37      static int rev[100];
38      int i,j=0;
39      for(i=arr_count-1;i>=0;i--)
40          rev[j++]=arr[i];
41      return rev;
42  }
43  }
44
42
43
44

```

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! ✓

```
1  /*
2  * Complete the 'cutThemAll' function below.
3  *
4  * The function is expected to return a STRING.
5  * The function accepts following parameters:
6  * 1. LONG_INTEGER_ARRAY lengths
7  * 2. LONG_INTEGER minLength
8  */
9
7  * 2. LONG_INTEGER minLength
8  */
9
10 /*
11 * To return the string from the function, you should either do stat
12 *
13 * For example,
14 * char* return_string_using_static_allocation() {
15 *     static char s[] = "static allocation of string";
16 *
```

```
13 * For example,
14 * char* return_string_using_static_allocation() {
15 *     static char s[] = "static allocation of string";
16 *
17 *     return s;
18 * }
19 *
20 * char* return_string_using_dynamic_allocation() {
21 *     char* s = malloc(100 * sizeof(char));
22 *
23 *     s = "dynamic allocation of string";
24 *
25 *     return s;
26 * }
27 *
28 */
```

```

27  *
28  */
29  char* cutThemAll(int lengths_count, long *lengths, long minLength) {
30      int s=0;
31      for(int i=0;i<lengths_count-1;i++)
32      {
33          s+=*(lengths+i);
34      }
35      if(s>=minLength)
36      {

```

```

32      {
33          s+=*(lengths+i);
34      }
35      if(s>=minLength)
36      {
37          return "Possible";
38      }
39      else
40      {
41          return "Impossible";
42      }
43  }
44  }
45

```

	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	✓