# Problem 2098: Subsequence of Size K With the Largest Even Sum

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

You are given an integer array

nums

and an integer

k

. Find the

largest even sum

of any subsequence of

nums

that has a length of

k

.

Return

this sum, or

-1

if such a sum does not exist

.

A

subsequence

is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

Input:

nums = [4,1,5,3,1], k = 3

Output:

12

Explanation:

The subsequence with the largest possible even sum is [4,5,3]. It has a sum of 4 + 5 + 3 = 12.

Example 2:

Input:

nums = [4,6,2], k = 3

Output:

12

Explanation:

The subsequence with the largest possible even sum is [4,6,2]. It has a sum of 4 + 6 + 2 = 12.

Example 3:

Input:

nums = [1,3,5], k = 1

Output:

-1

Explanation:

No subsequence of nums with length 1 has an even sum.

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 10

5

1 <= k <= nums.length

## Code Snippets

**C++:**

```
class Solution {
public:
long long largestEvenSum(vector<int>& nums, int k) {

}
};
```

**Java:**

```
class Solution {
public long largestEvenSum(int[] nums, int k) {

}
}
```

**Python3:**

```
class Solution:
def largestEvenSum(self, nums: List[int], k: int) -> int:
```

**Python:**

```
class Solution(object):
def largestEvenSum(self, nums, k):
"""
:type nums: List[int]
:type k: int
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[]} nums
* @param {number} k
* @return {number}
*/
var largestEvenSum = function(nums, k) {

};
```

**TypeScript:**

```
function largestEvenSum(nums: number[], k: number): number {

};
```

**C#:**

```
public class Solution {
public long LargestEvenSum(int[] nums, int k) {

}
}
```

**C:**

```
long long largestEvenSum(int* nums, int numsSize, int k) {

}
```

**Go:**

```
func largestEvenSum(nums []int, k int) int64 {

}
```

**Kotlin:**

```
class Solution {
fun largestEvenSum(nums: IntArray, k: Int): Long {

}
}
```

**Swift:**

```
class Solution {
func largestEvenSum(_ nums: [Int], _ k: Int) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn largest_even_sum(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

**Ruby:**

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def largest_even_sum(nums, k)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function largestEvenSum($nums, $k) {

}
}
```

**Dart:**

```
class Solution {
int largestEvenSum(List<int> nums, int k) {

}
}
```

**Scala:**

```
object Solution {
def largestEvenSum(nums: Array[Int], k: Int): Long = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec largest_even_sum(nums :: [integer], k :: integer) :: integer
def largest_even_sum(nums, k) do
```

```
end
end
```

### Erlang:

```
-spec largest_even_sum(Nums :: [integer()], K :: integer()) -> integer().
largest_even_sum(Nums, K) ->
.
```

### Racket:

```
(define/contract (largest-even-sum nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Subsequence of Size K With the Largest Even Sum
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
long long largestEvenSum(vector<int>& nums, int k) {

}
};
```

### Java Solution:

```
/**
* Problem: Subsequence of Size K With the Largest Even Sum
```

```
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public long largestEvenSum(int[] nums, int k) {

}
}
```

### Python3 Solution:

```
"""
Problem: Subsequence of Size K With the Largest Even Sum
Difficulty: Medium
Tags: array, greedy, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def largestEvenSum(self, nums: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def largestEvenSum(self, nums, k):
"""
:type nums: List[int]
:type k: int
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Subsequence of Size K With the Largest Even Sum
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @param {number} k
* @return {number}
*/
var largestEvenSum = function(nums, k) {

};
```

### TypeScript Solution:

```
/**
* Problem: Subsequence of Size K With the Largest Even Sum
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function largestEvenSum(nums: number[], k: number): number {

};
```

### C# Solution:

```
/*
* Problem: Subsequence of Size K With the Largest Even Sum
* Difficulty: Medium
* Tags: array, greedy, sort
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public long LargestEvenSum(int[] nums, int k) {

}
}
```

### C Solution:

```
/*
* Problem: Subsequence of Size K With the Largest Even Sum
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

long long largestEvenSum(int* nums, int numsSize, int k) {

}
```

### Go Solution:

```
// Problem: Subsequence of Size K With the Largest Even Sum
// Difficulty: Medium
// Tags: array, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func largestEvenSum(nums []int, k int) int64 {

}
```

### Kotlin Solution:

```
class Solution {
fun largestEvenSum(nums: IntArray, k: Int): Long {

}
}
```

### Swift Solution:

```
class Solution {
func largestEvenSum(_ nums: [Int], _ k: Int) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Subsequence of Size K With the Largest Even Sum
// Difficulty: Medium
// Tags: array, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn largest_even_sum(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

### Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def largest_even_sum(nums, k)

end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function largestEvenSum($nums, $k) {

}
}
```

### Dart Solution:

```
class Solution {
int largestEvenSum(List<int> nums, int k) {

}
}
```

### Scala Solution:

```
object Solution {
def largestEvenSum(nums: Array[Int], k: Int): Long = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec largest_even_sum(nums :: [integer], k :: integer) :: integer
def largest_even_sum(nums, k) do

end
end
```

### Erlang Solution:

```
-spec largest_even_sum(Nums :: [integer()], K :: integer()) -> integer().
largest_even_sum(Nums, K) ->
.
```

**Racket Solution:**

```
(define/contract (largest-even-sum nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```