

26) Given a circular integer array `nums` of length `n`, return the maximum possible sum of a non-empty subarray of `nums`. A circular array means the end of the array connects to the beginning of the array. Formally, the next element of `nums[i]` is `nums[(i + 1) % n]` and the previous element of `nums[i]` is `nums[(i - 1 + n) % n]`. A subarray may only include each element of the fixed buffer `nums` at most once. Formally, for a subarray `nums[i], nums[i + 1], ..., nums[j]`, there does not exist $i \leq k_1, k_2 \leq j$ with $k_1 \% n == k_2 \% n$.

CODE:

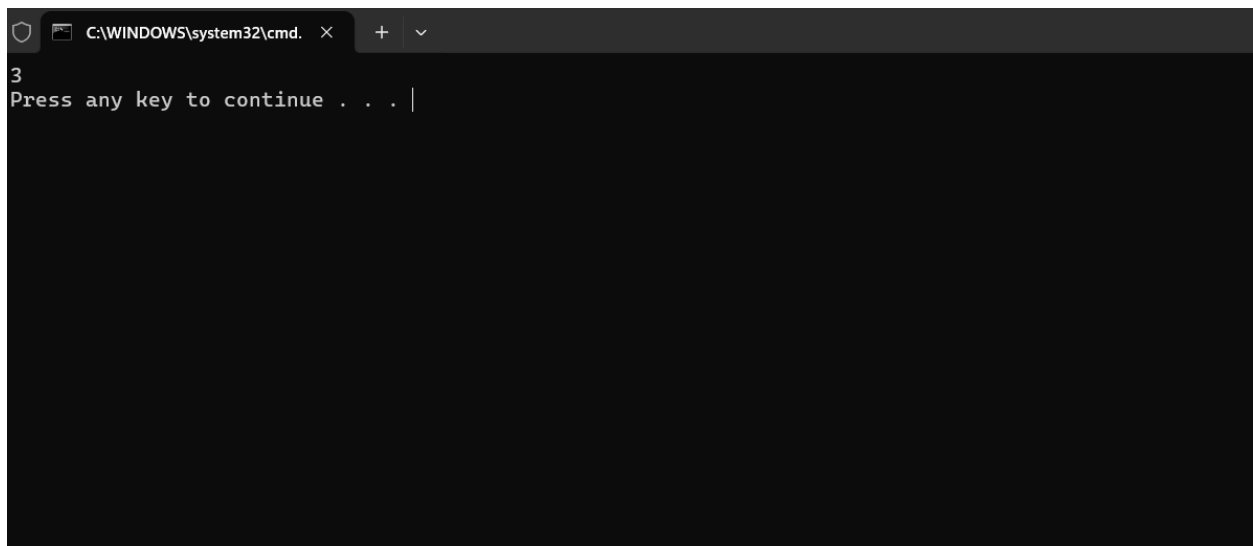
```
def maxSubarraySumCircular(nums):
    def kadane(arr):
        max_sum = float('-inf')
        curr_sum = 0
        for num in arr:
            curr_sum = max(num, curr_sum + num)
            max_sum = max(max_sum, curr_sum)
        return max_sum

    total_sum = sum(nums)
    max_standard = kadane(nums)
    max_wrap = total_sum + kadane([-num for num in nums])

    return max(max_standard, max_wrap) if max_wrap != 0 else max_standard

nums = [1, -2, 3, -2]
result = maxSubarraySumCircular(nums)
print(result)
```

OUTPUT:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the number '3' on the first line, followed by the text 'Press any key to continue . . . |' on the second line. The rest of the window is empty.

TIME COMPLEXITY : $O(n)$