LONGEST PEAK SUBARRAY

```
def longest_peak_subarray(arr):
    n = len(arr)
    if n < 3:
        return 0
    max_len = 0
        i = 1
     while i < n - 1:
                if arr[i - 1] < arr[i] > arr[i + 1]:
                    left = i - 1
                     right = i + 1

         while left > 0 and arr[left - 1] < arr[left]:
                      left -= 1

          while right < n - 1 and arr[right] > arr[right + 1]:
                         right += 1



           peak = arr[i]
           is_unique_peak = True
           for j in range(left, right + 1):
               if j != i and arr[j] == peak:
```

```
                is_unique_peak = False
                break

        if is_unique_peak:
            max_len = max(max_len, right - left + 1)

        i = right
    else:
        i += 1

return max_len

n = int(input())
arr = list(map(int, input().split()))
print(longest_peak_subarray(arr))
```

THE FORGOTTEN SONG 1

```
def longest_repeating_substring(s):
    def is_valid(mid):
        seen = set()
        for i in range(len(s) - mid + 1):
            substring = s[i:i + mid]
```

```
            if substring in seen:
                return substring
            seen.add(substring)
        return None

    left, right = 0, len(s)
    result = "-1"

    while left <= right:
        mid = (left + right) // 2
        found = is_valid(mid)
        if found:
            result = found
            left = mid + 1
        else:
            right = mid - 1

    return result


s = input().strip()
print(longest_repeating_substring(s))
```

THE LOST TREASURE TRAIL

```
def max_subarray_sum(arr):
 max_sum = float('-inf')
 current_sum = 0
 max_length = 0
 current_length = 0

 for num in arr:
   current_sum += num
   current_length += 1

   if current_sum > max_sum:
      max_sum = current_sum
      max_length = current_length

   if current_sum < 0:
      current_sum = 0
      current_length = 0

 return max_sum
n=int(input())
```

```
arr=list(map(int,input().split()))
print(max_subarray_sum(arr))
```

THE SECRET CODE

```
def min_window(S, T):
    def contains(window, T):
        for char in T:
            if char not in window or window.count(char) < T.count(char):
                return False
        return True

    min_len = float('inf')
    min_window = ""

    for i in range(len(S)):
        for j in range(i + 1, len(S) + 1):
            window = S[i:j]
            if contains(window, T) and len(window) < min_len:
                min_len = len(window)
                min_window = window
```

```
    return min_window

S = input()
T = input()

print( min_window(S, T))
```

THE JUMBLED INVITATIONS

```
def are_anagrams(str1, str2):

    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    return sorted(str1) == sorted(str2)

str1 = input()
str2 = input()

if are_anagrams(str1, str2):
```

```
    print("True")
else:
    print("False")
```