```
#find missing number
def find_missing_number(nums):
    n = len(nums) + 1  # The actual size should be n+1
    expected_sum = n * (n + 1) // 2  # Sum of first n natural numbers
    actual_sum = sum(nums)  # Sum of elements in given list
    return expected_sum - actual_sum  # The missing number

# Example usage
nums = [1, 2, 4, 5, 6]  # Missing number should be 3
print("Missing Number:", find_missing_number(nums))
```

#Check Balanced Parentheses

```
def is_balanced(s):
    stack = []
    pairs = {')': '(', '}': '{', ']': '['}  # Mapping of closing to opening brackets

    for char in s:
        if char in "({[":
            stack.append(char)  # Push opening bracket onto stack
        elif char in ")}]":
            if not stack or stack.pop() != pairs[char]:  # Mismatch or empty stack
                return False

    return not stack  # If stack is empty, it's balanced

# Example usage
expr = input("Enter a string of parentheses: ")
print("Balanced:", is_balanced(expr))
```

```
#longest word in the sentence
def longest_word(sentence):
    words = sentence.split()  # Split sentence into words
    return max(words, key=len) if words else None  # Find the longest word

# Example usage
sentence = input("Enter a sentence: ")
print("Longest word:", longest_word(sentence))



#count word in a sentence


def count_words(sentence):
    return len(sentence.split())  # Split by spaces and count words

# Example usage
sentence = input("Enter a sentence: ")
print("Word count:", count_words(sentence))
```

#Check Pythagorean Triplet

```
def is_pythagorean_triplet(a, b, c):
    # Sort the numbers to ensure c is the largest
    x, y, z = sorted([a, b, c])
    return x**2 + y**2 == z**2  # Check Pythagorean theorem

# Example usage
a, b, c = map(int, input("Enter three numbers: ").split())
```

```
print("Pythagorean Triplet:", is_pythagorean_triplet(a, b, c))



#bubble sort

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        swapped = False  # Track if any swaps occur
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:  # Swap if elements are out of order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:  # If no swaps in a pass, array is already sorted
            break
    return arr

# Example usage
arr = list(map(int, input("Enter numbers separated by space: ").split()))
print("Sorted List:", bubble_sort(arr))



#binary search

def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
```

```
        mid = (left + right) // 2  # Find the middle index

        if arr[mid] == target:
            return mid  # Target found
        elif arr[mid] < target:
            left = mid + 1  # Search right half
        else:
            right = mid - 1  # Search left half

    return -1  # Target not found

# Example usage
arr = list(map(int, input("Enter sorted numbers: ").split()))
target = int(input("Enter target: "))
result = binary_search(arr, target)
print("Index:", result if result != -1 else "Not Found")



#find subarray in given sum

def find_subarray_with_sum(arr, S):
    left, curr_sum = 0, 0

    for right in range(len(arr)):
        curr_sum += arr[right]  # Expand window

        while curr_sum > S and left <= right:
            curr_sum -= arr[left]  # Shrink window
            left += 1
```

```
        if curr_sum == S:
            return (left, right)  # Return indices

    return -1  # No subarray found

# Example usage
arr = list(map(int, input("Enter numbers: ").split()))
S = int(input("Enter target sum: "))
result = find_subarray_with_sum(arr, S)
print("Subarray indices:", result if result != -1 else "Not Found")
```