```
def find_missing_number():
    n = int(input("Enter n: "))
    arr = list(map(int, input("Enter numbers: ").split()))
    expected_sum = (n * (n + 1)) // 2
    actual_sum = sum(arr)
    return expected_sum - actual_sum
print(find_missing_number())
```

```
Enter n: 4
Enter numbers: 1 2 4 5
-2
```

```
def is_balanced():
    s = input("Enter parentheses string: ")
    stack = []
    mapping = {')': '(', '}': '{', ']': '['}
    for char in s:
        if char in mapping:
            top_element = stack.pop() if stack else '#'
            if mapping[char] != top_element:
                return False
        else:
            stack.append(char)
    return not stack
print(is_balanced())
```

```
Enter parentheses string: bhuvan
False
```

```
def longest_word():
    sentence = input("Enter a sentence: ")
    words = sentence.split()
    return max(words, key=len)
print(longest_word())
```

```
Enter a sentence: bhuvan is good
bhuvan
```

```
def count_words():
    sentence = input("Enter a sentence: ")
    return len(sentence.split())
print(count_words())
```

```
Enter a sentence: bhuvan is bad
3
```

```
def is_pythagorean_triplet():
    a, b, c = map(int, input("Enter three numbers: ").split())
    x, y, z = sorted([a, b, c])
    return x*x + y*y == z*z
print(is_pythagorean_triplet())
```

```
Enter three numbers: 1 2 3
False
```

```
def bubble_sort():
    arr = list(map(int, input("Enter numbers: ").split()))
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
print(bubble_sort())
```

```
Enter numbers: 2 4 1
[1, 2, 4]
```

```
def binary_search():
    arr = list(map(int, input("Enter sorted numbers: ").split()))
    target = int(input("Enter target: "))
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
print(binary_search())
```

```
Enter sorted numbers: 1 2 3 4 5 6 7
Enter target: 2
1
```

```
def find_subarray_with_sum():
    arr = list(map(int, input("Enter numbers: ").split()))
    S = int(input("Enter sum: "))
    left, current_sum = 0, 0
    for right in range(len(arr)):
        current_sum += arr[right]
        while current_sum > S and left <= right:
            current_sum -= arr[left]
            left += 1
        if current_sum == S:
            return (left, right)
    return -1
print(find_subarray_with_sum())
```

```
Enter numbers: 1 2 3 4 5
Enter sum: 15
(0, 4)
```

```
from collections import Counter

def log_analysis():
    # Simulating log data since we can't upload a file
    logs = [
        "192.168.1.1 200 /home",
        "192.168.1.2 404 /about",
        "192.168.1.1 200 /home",
        "192.168.1.3 500 /contact",
        "192.168.1.1 200 /home",
        "192.168.1.4 200 /services",
        "192.168.1.2 404 /about",
        "192.168.1.1 200 /home",
        "192.168.1.3 500 /contact",
        "192.168.1.5 302 /redirect"
    ]

    ip_counter = Counter()
    url_counter = Counter()
    status_counter = Counter()

    for log in logs:
        parts = log.split()
        if len(parts) < 3:
            continue
        ip, status, url = parts[0], parts[1], parts[2]
        ip_counter[ip] += 1
        url_counter[url] += 1
        status_counter[status] += 1

    result = {
        "Most Frequent IPs": ip_counter.most_common(5),
        "Most Accessed URLs": url_counter.most_common(5),
        "Response Code Counts": status_counter.most_common(5)
    }

    # Print results
    for key, value in result.items():
        print(f"{key}: {value}")

log_analysis()
```

```
Most Frequent IPs: [('192.168.1.1', 4), ('192.168.1.2', 2), ('192.168.1.3', 2), ('192.168.1.4', 1), ('192.168.1.5', 1)]
Most Accessed URLs: [('/home', 4), ('/about', 2), ('/contact', 2), ('/services', 1), ('/redirect', 1)]
Response Code Counts: [('200', 5), ('404', 2), ('500', 2), ('302', 1)]
```