```python
#1. Parentheses Positioning Error
def find_mismatch_position(s):
    stack = []
    for i, char in enumerate(s):
        if char == '(':
            stack.append(i)
        elif char == ')':
            if not stack:
                return i
            stack.pop()
    return stack[0] if stack else -1
string = "(()())(()("
print("Mismatched position:", find_mismatch_position(string))
```

```
Mismatched position: 8
```

```python
#2. Parentheses in a Long String
def balanced_parentheses_positions(s):
    stack = []
    balanced_positions = []
    for i, char in enumerate(s):
        if char == '(':
            stack.append(i)
        elif char == ')':
            if stack:
                balanced_positions.append((stack.pop(), i))
    return balanced_positions


string = "(()())(()("
print("Balanced positions:", balanced_parentheses_positions(string))
```

```
Balanced positions: [(2, 3), (1, 4), (5, 6), (0, 7), (9, 10)]
```

```python
#3. Minimum Number of Parentheses to Add
def min_parentheses_to_add(s):
    open_needed = 0
    close_needed = 0
    for char in s:
        if char == '(':
            close_needed += 1
        elif char == ')':
            if close_needed > 0:
                close_needed -= 1
            else:
                open_needed += 1
    return open_needed + close_needed

string = "(()())(()("
print("Minimum parentheses to add:", min_parentheses_to_add(string))
```

```
Minimum parentheses to add: 1
```

```python
#4. Longest Valid Parentheses Substring
def longest_valid_parentheses(s):
    stack = [-1]
    max_length = 0
    for i, char in enumerate(s):
        if char == '(':
            stack.append(i)
        elif char == ')':
            stack.pop()
            if not stack:
                stack.append(i)
            else:
                max_length = max(max_length, i - stack[-1])
    return max_length


string = "((()))(())(()"
print("Longest valid parentheses length:",
longest_valid_parentheses(string))
```

```
Longest valid parentheses length: 8
```