Programming Solutions

1. Letter-Frequency Histogram

```python
def letter_histogram(text):
    counts = {}
    for ch in text:
        if ch.isalpha():
            ch_low = ch.lower()
            counts[ch_low] = counts.get(ch_low, 0) + 1

    items = list(counts.items())

    def sort_key(pair):
        # pair = (letter, count)
        return (-pair[1], pair[0])

    items.sort(key=sort_key)

    for letter, count in items:
        print(letter + " " + str(count))

print("Sample 1:")
letter_histogram("Hello, World!")
print("\nSample 2:")
letter_histogram("banana BANANA")
```

2. Run-Length Encoder / Decoder

```python
def compress(s):
    if not s:
        return ""
    out = []
    current = s[0]
    run = 1
    for ch in s[1:]:
        if ch == current and run < 9:
            run += 1
        else:
            out.append(current + str(run))
```

```
            current = ch
            run = 1
    out.append(current + str(run))
    return "".join(out)


def decompress(rle):
    out = []
    i = 0
    while i < len(rle):
        ch = rle[i]
        cnt = int(rle[i + 1])
        out.append(ch * cnt)
        i += 2
    return "".join(out)


for original in ["aaabbc", "abcd", "xxxxxxxxxxxy"]:
    encoded = compress(original)
    decoded = decompress(encoded)
    print(original, "->", encoded, "->", decoded)
```

3. Print Matrix in Spiral Order

```
def spiral_print(n):
    grid = [[0] * n for _ in range(n)]
    top = 0
    bottom = n - 1
    left = 0
    right = n - 1
    num = 1

    while top <= bottom and left <= right:
        # move right
        for col in range(left, right + 1):
            grid[top][col] = num
            num += 1
        top += 1

        # move down
        for row in range(top, bottom + 1):
            grid[row][right] = num
            num += 1
        right -= 1

        if top <= bottom:
            # move left
            for col in range(right, left - 1, -1):
                grid[bottom][col] = num
                num += 1
```

```
            bottom -= 1

        if left <= right:
            # move up
            for row in range(bottom, top - 1, -1):
                grid[row][left] = num
                num += 1
            left += 1

    return grid


for n in (3, 4, 5):
    print("\nSpiral for n =", n)
    for row in spiral_print(n):
        print(row)
```

4. Tic-Tac-Toe Winner

```
def winner(board):
    lines = []

    # rows and columns
    for i in range(3):
        lines.append(board[i])
        column = [board[r][i] for r in range(3)]
        lines.append(column)

    # diagonals
    diag1 = [board[i][i] for i in range(3)]
    diag2 = [board[i][2 - i] for i in range(3)]
    lines.append(diag1)
    lines.append(diag2)

    for line in lines:
        if line[0] != "" and line.count(line[0]) == 3:
            return line[0]
    return None


tests = [
    [['X','O','X'],['O','X','O'],['','O','X']],
    [['O','O','O'],['X','','X'],['','X','']],
    [['X','O','X'],['O','X','O'],['O','X','O']]
]
for b in tests:
    print(b, "->", winner(b))
```

5. Merge Overlapping Intervals

```python
def start_key(interval):
    return interval[0]


def merge(intervals):
    if not intervals:
        return []

    intervals.sort(key=start_key)
    merged = [intervals[0]]

    for start, end in intervals[1:]:
        last_start, last_end = merged[-1]
        if start <= last_end:  # overlap or touch
            merged[-1] = (last_start, max(last_end, end))
        else:
            merged.append((start, end))
    return merged


examples = [
    [(1, 3), (2, 4), (5, 7), (7, 8)],
    [(0, 1), (2, 3)]
]
for ex in examples:
    print(ex, "->", merge(ex))
```

6. Balanced Brackets

```python
def is_balanced(expr):
    depth = 0
    for ch in expr:
        if ch == '(':
            depth += 1
        else:  # ch == ')'
            depth -= 1
            if depth < 0:
                return False
    return depth == 0


samples = ["(())", "(()())", "())(", "(()"]
for s in samples:
    print(s, "->", is_balanced(s))
```