Batch:A3 **Experiment Number: 6**

Roll Number:16010423099 **Name: Suryanshu Banerjee**

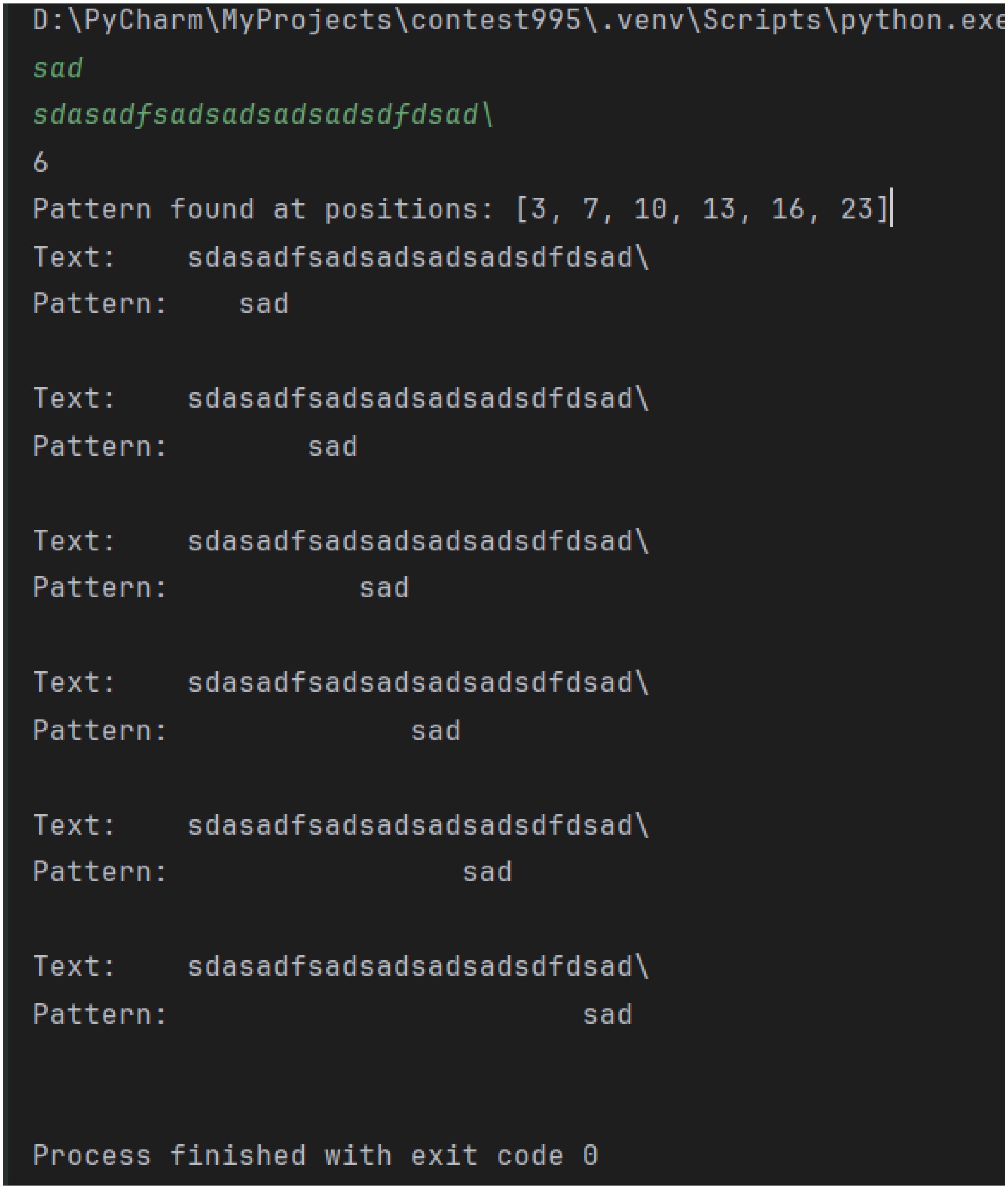
Aim of the Experiment: To study implement KMP Algorithm

Program/ Steps:

```
def compute_lps_array(pattern):
    m = len(pattern)
    lps = [0] * m

    length = 0
    i = 1

    while i < m:
        if pattern[i] == pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length - 1]
            else:
                lps[i] = 0
                i += 1

    return lps
```

```
def kmp_search(pattern, text):
    if not pattern or not text:
        return 0

    count = 0
    found_positions = []  # Store positions where pattern is found

    m = len(pattern)
    n = len(text)

    lps = compute_lps_array(pattern)

    i = 0  # text position
    j = 0  # pattern position

    while i < n:
        # Characters match
        if pattern[j] == text[i]:
            i += 1
            j += 1

        # Pattern found
        if j == m:
            position = i - j  # Calculate where pattern starts
            found_positions.append(position)
            count += 1
            j = lps[j - 1]

        # Mismatch after j matches
        elif i < n and pattern[j] != text[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1

    return count, found_positions



def main():
    # Read input
    pattern = input().strip()
    text = input().strip()

    # Find occurrences using KMP
    count, positions = kmp_search(pattern, text)

    # Print total count
    print(count)

    # Print where patterns were found
    if count > 0:
        print(f"Pattern found at positions: {positions}")
        # Show visual representation
        for pos in positions:
            print(f"Text:    {text}")
            print(f"Pattern: {' ' * pos}{pattern}")
            print()


if __name__ == "__main__":
    main()
```

Output/Result:



```
D:\PyCharm\MyProjects\contest995\.venv\Scripts\python.exe
sad
sdasadfsadsadsadsadsdfdsad\
6
Pattern found at positions: [3, 7, 10, 13, 16, 23]
Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:    sad

Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:        sad

Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:           sad

Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:              sad

Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:                 sad

Text:    sdasadfsadsadsadsadsdfdsad\
Pattern:                        sad


Process finished with exit code 0
```

Post Lab Question-Answers:

None.

Outcomes:

Understand the Graphs, related algorithms, efficient implementation of those algorithms and applications.

Conclusion (based on the Results and outcomes achieved):

Successfully implemented KMP algorithm to find occurrences of string substring.

References:

1. https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/algorithm/sum-as-per-frequency-88b00c1f/
2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
3. Antti Laaksonen, “Guide to Competitive Programming”,Springer,2018
4. Gayle Laakmann McDowell,” Cracking the Coding Interview”,CareerCup LLC,2015
5. Steven S. Skiena Miguel A. Revilla,”Programming challenges, The Programming Contest Training Manual”, Springer, 2006
6. Antti Laaksonen, “Competitive Programmer’s Handbook”, Hand book, 2018
7. Steven Halim and Felix Halim, “Competitive Programming 3: The Lower Bounds of Programming Contests”, Handbook for ACM ICPC