BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	Balla Mahadev Shrikrishna
UID no.	2023300010
Experiment No.	3

AIM:	To implement String Matching Algorithm.
Program 1	
PROBLEM STATEMENT :	**Program Implementation:** Implement a Rabin Karp Algorithm as function which takes input text (i.e. T) as an array of Minimum of 1K or Maximum of 10K characters and pattern text (i.e. P) as an array of 10,20,…,100 characters. You may need to use some buffer management scheme if sufficient storage (e.g. 10K) is not available in Main Memory of the OS. **Input:** 1) Each student has to create 10 text files of input sizes 1K, 2K,…,10K using one of the kaggle datasets e.g. Big Text or Random Text given in the Important Links section. 2) Input 10 pattern texts (i.e. P) as an array of 10,20,…,100 characters [Some are manual and some randomly generated from any tool]. Some of the input pattern must be spurious and some must be actual. 3) Use efficient input, output operations are encouraged for reading these 10 files. **Output:** 1) Print the time required to search 10 patterns for 10 input files [total 100 combinations] 2) Plot these time required to search 10 patterns for 10 input files as XY plot where a. X represents input pattern sizes (i.e. P). b. Y represents time taken for searching patterns in the input text files c. Each line represents the input Text File (i.e. T)

PROGRAM (rabinkarp.cpp):

```
#include <bits/stdc++.h>
using namespace std;
using namespace chrono;

#define d 256  // Base for hash computation (256 for ASCII characters, 10 for nos.)
#define q 101  // Prime number for modular arithmetic to reduce hash collisions

const int NUM_FILES = 10;   // Number of input text files
const int PATTERN_COUNT = 10; // Number of patterns per file
const vector<int> PATTERN_SIZES = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 }; // Pattern sizes

vector<string> input_files(NUM_FILES); // Stores file names

// Function to pre-process text (remove spaces, newlines, and normalize)
string preprocess_text(const string& text) {
        string processed;
        for (char c : text) {
        if (c == ' ' || c == '\n' || c == '\t' || c == '\r') {
        continue; // Skip spaces, newlines, and tabs
        }
        processed += tolower(c); // Convert to lowercase
        }
        return processed;
}

// Function to generate and save text files with varying sizes (1K to 10K characters)
void generate_text_files() {
        ifstream kaggle_file("big.txt"); // file extracted from Kaggle dataset
        if (!kaggle_file) {
        cerr << "Error opening Kaggle dataset file!" << endl;
        return;
        }

        string text;
```

```
    getline(kaggle_file, text, '\0'); // Read entire dataset
    kaggle_file.close();

    if (text.size() < 10000) {
    cerr << "Dataset too small!" << endl;
    return;
    }

    for (int i = 1; i <= NUM_FILES; i++) {
    int size = i * 1000; // 1K to 10K
    string filename = "input_" + to_string(size) + ".txt";
    input_files[i - 1] = filename;

    ofstream file(filename);
    file << text.substr(0, size); // Save only first 'size' characters
    file.close();
    }
}

// Function to generate random or real patterns
vector<string> generate_patterns(const string& text) {
    vector<string> patterns;
    for (int size : PATTERN_SIZES) {
    if (rand() % 2 == 0) {
    // Real pattern: Extract a substring of given size from the text
    int start = rand() % (text.size() - size);
    patterns.push_back(text.substr(start, size));
    }
    else {
    // Spurious pattern: Generate a random string of given size
    string spurious(size, ' ');
    for (char& c : spurious) {
        c = 'A' + rand() % 26; // Random uppercase letter
    }
    patterns.push_back(spurious);
    }
    }
    return patterns;
```

```
}

// Rabin-Karp algorithm
int rabin_karp_matcher(const string& T, const string& P) {
        int n = T.length(); // Length of the text
        int m = P.length(); // Length of the pattern
        int h = 1; // Hash multiplier for rolling hash
        int p = 0; // Hash value for the pattern
        int t = 0; // Hash value for the current window in the text
        int match_count = 0; // Count of pattern matches in the text

        // Calculate h = d^(m-1) % q
        for (int i = 0; i < m - 1; i++) {
        h = (h * d) % q;
        }

        // Calculate initial hash values for the pattern and the first window of
the text
        for (int i = 0; i < m; i++) {
        p = (d * p + P[i]) % q;
        t = (d * t + T[i]) % q;
        }

        // Slide the pattern over the text one character at a time
        for (int s = 0; s <= n - m; s++) {
        // If hash values match, check character by character
        if (p == t) {
        if (T.substr(s, m) == P) {
                match_count++; // Increment match count if pattern matches
        }
        }

        // Calculate hash value for the next window
        if (s < n - m) {
        t = (d * (t - T[s] * h) + T[s + m]) % q;
        if (t < 0) t += q; // Ensure hash value is non-negative
        }
        }
        return match_count;
}

// Main execution
int main() {
        srand(time(0)); // Seed random number generator
        generate_text_files(); // Generate input files from Kaggle dataset

        ofstream log_file("timing_results.csv"); // Log file for timing results
        log_file <<
"PatternSize,InputFile,Text(Ts),Pattern(Ps),TimeTaken(ms)\n"; // CSV
header

        for (const string& filename : input_files) {
        ifstream file(filename);
        if (!file) {
        cerr << "Error opening " << filename << endl;
        continue;
        }

        // Read the entire text file into memory
        string text;
        getline(file, text, '\0');
        file.close();

        // Pre-process the text (remove spaces, newlines, and normalize)
        string processed_text = preprocess_text(text);

        // Generate 10 patterns (real and spurious)
        vector<string> patterns = generate_patterns(processed_text);

        // Search for each pattern in the text and measure time
        for (const string& pattern : patterns) {
        auto start = high_resolution_clock::now(); // Start timer
        int matches = rabin_karp_matcher(processed_text, pattern); //
Perform pattern matching
        auto stop = high_resolution_clock::now(); // Stop timer
```

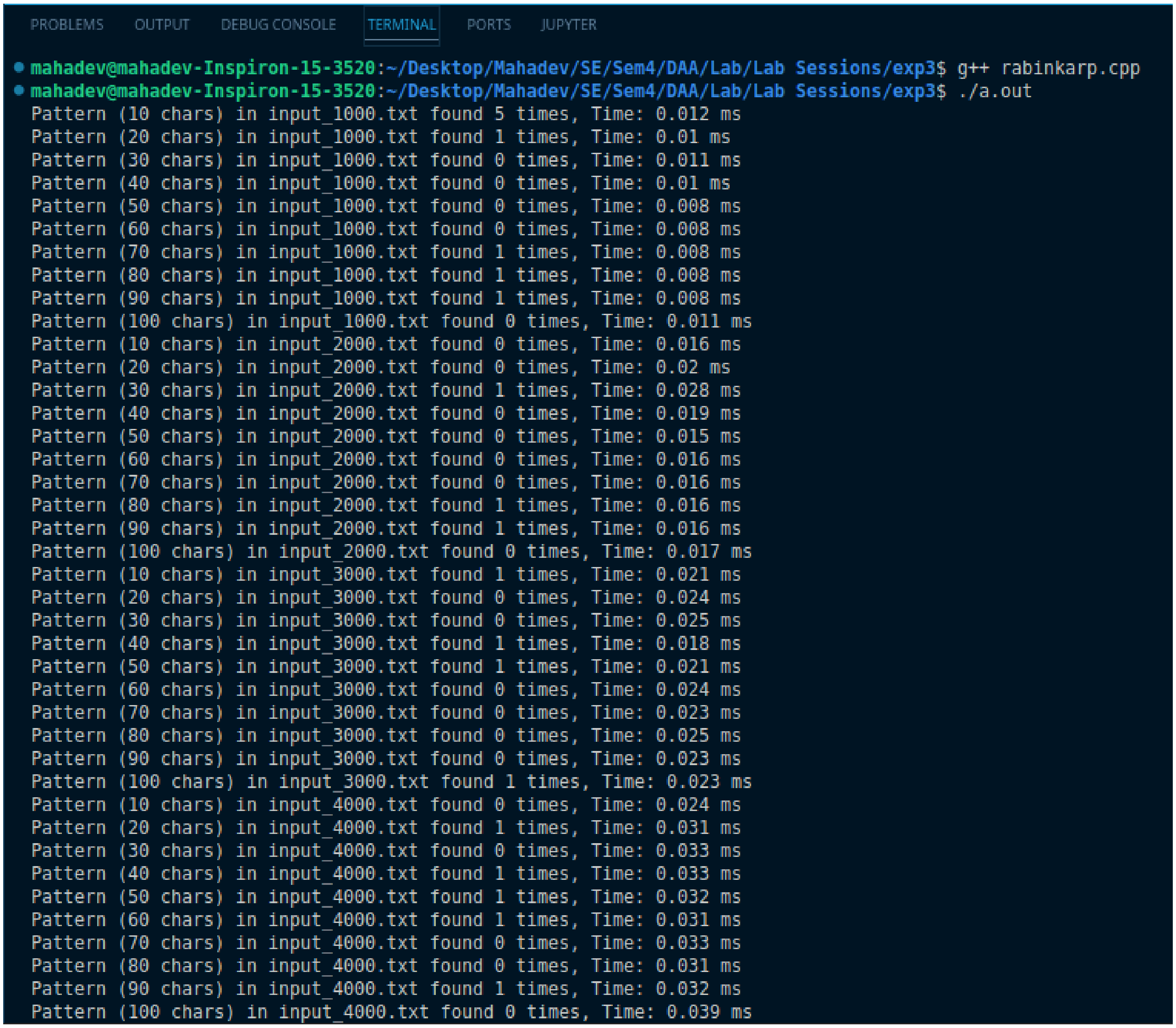
```
    // Calculate duration in milliseconds
    double duration = duration_cast<microseconds>(stop - start).count() / 1000.0;

    // Log results to CSV file
    log_file << pattern.length() << ","
        << filename << ","
        << "\"" << processed_text << "\","
        << "\"" << pattern << "\","
        << duration << "\n";

    // Print results to console
    cout << "Pattern (" << pattern.length() << " chars) in " << filename
        << " found " << matches << " times, Time: " << duration << " ms\n";
    }
    }

    log_file.close(); // Close log file
    return 0;
}
```

plot.ipynb:

```
import matplotlib.pyplot as plt
import pandas as pd

# Read the CSV file
try:
    data = pd.read_csv("timing_results.csv", quotechar='"', escapechar='\\')
except FileNotFoundError:
    print("Error: 'timing_results.csv' not found. Please run the C++ program first.")
    exit(1)
except pd.errors.ParserError:
    print("Error: Unable to parse 'timing_results.csv'. Ensure the file is properly formatted.")
    exit(1)
```

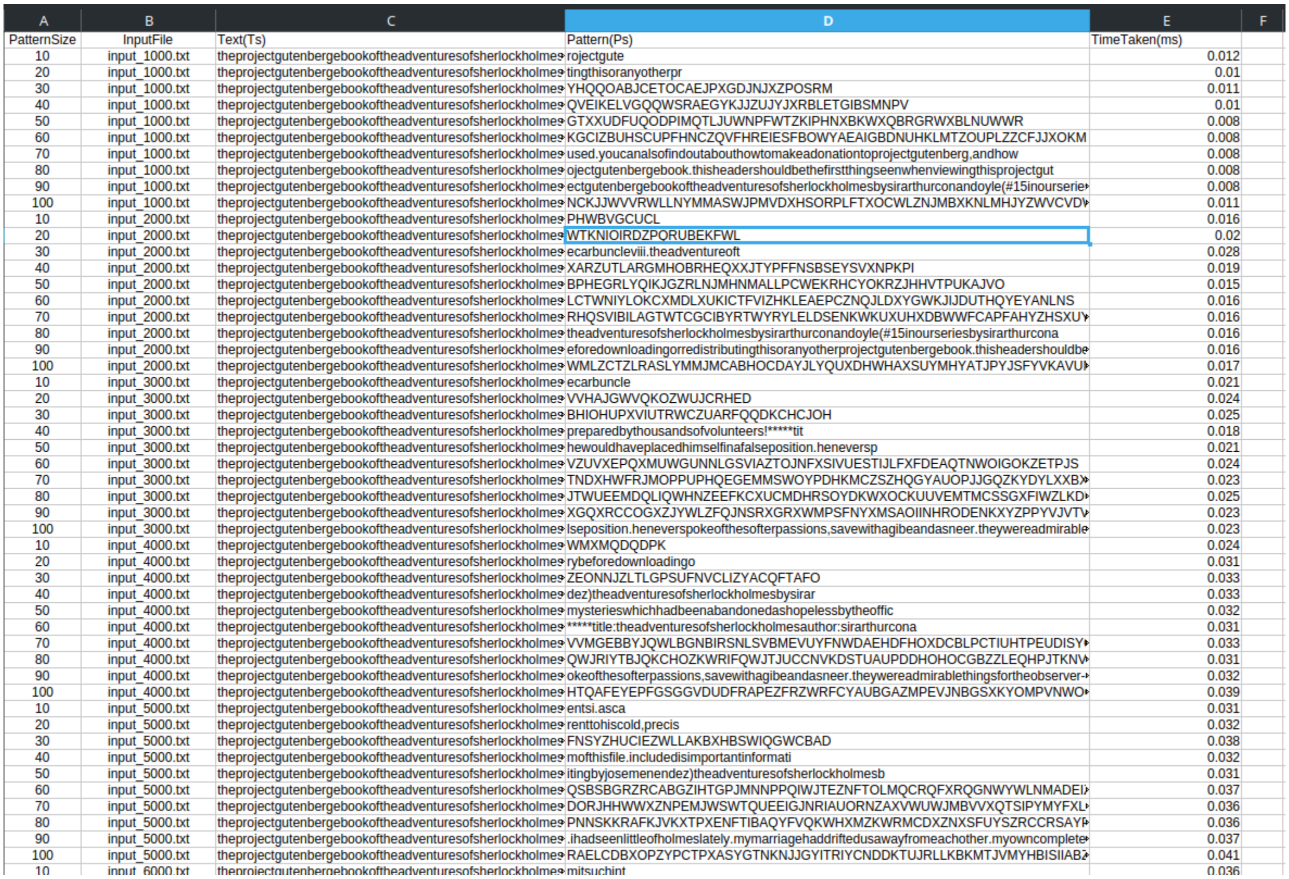
```
# Plot the data
plt.figure(figsize=(12, 8))  # Set figure size for better readability

# Plot a line for each input file
for file in data["InputFile"].unique():
    subset = data[data["InputFile"] == file]  # Filter data for the current file
    plt.plot(subset["PatternSize"], subset["TimeTaken(ms)"], label=file, marker='o')

# Add labels and title
plt.xlabel("Pattern Size (Characters)", fontsize=12)
plt.ylabel("Time Taken (ms)", fontsize=12)
plt.title("Pattern Size vs Time Taken for Different Input Files", fontsize=14)

# Add a legend
plt.legend(title="Input File", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=10)

# Add grid for better readability
plt.grid(True, linestyle='--', alpha=0.6)

# Display the plot
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()

# Save the plot as an image (optional)
plt.savefig("pattern_vs_time_plot.png", dpi=300, bbox_inches="tight")
```

RESULT:

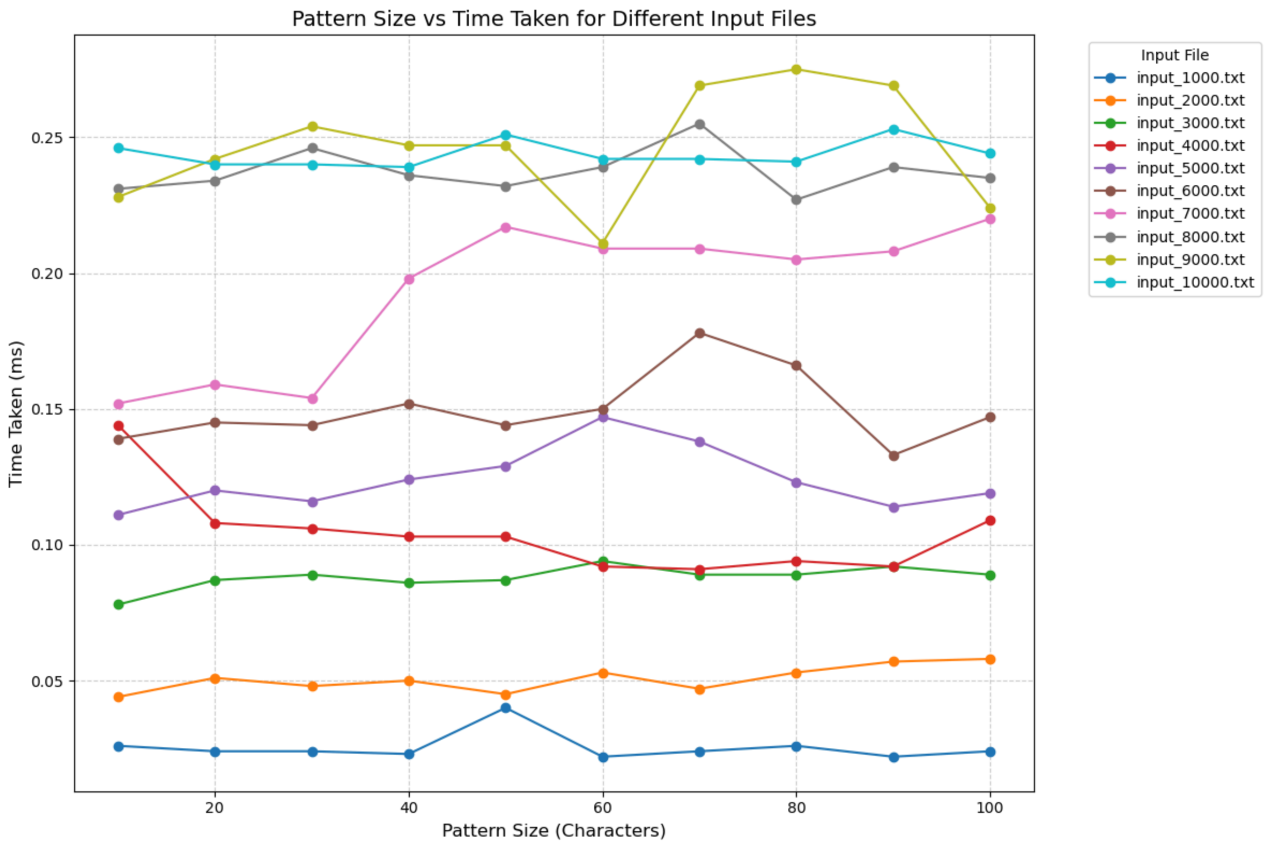


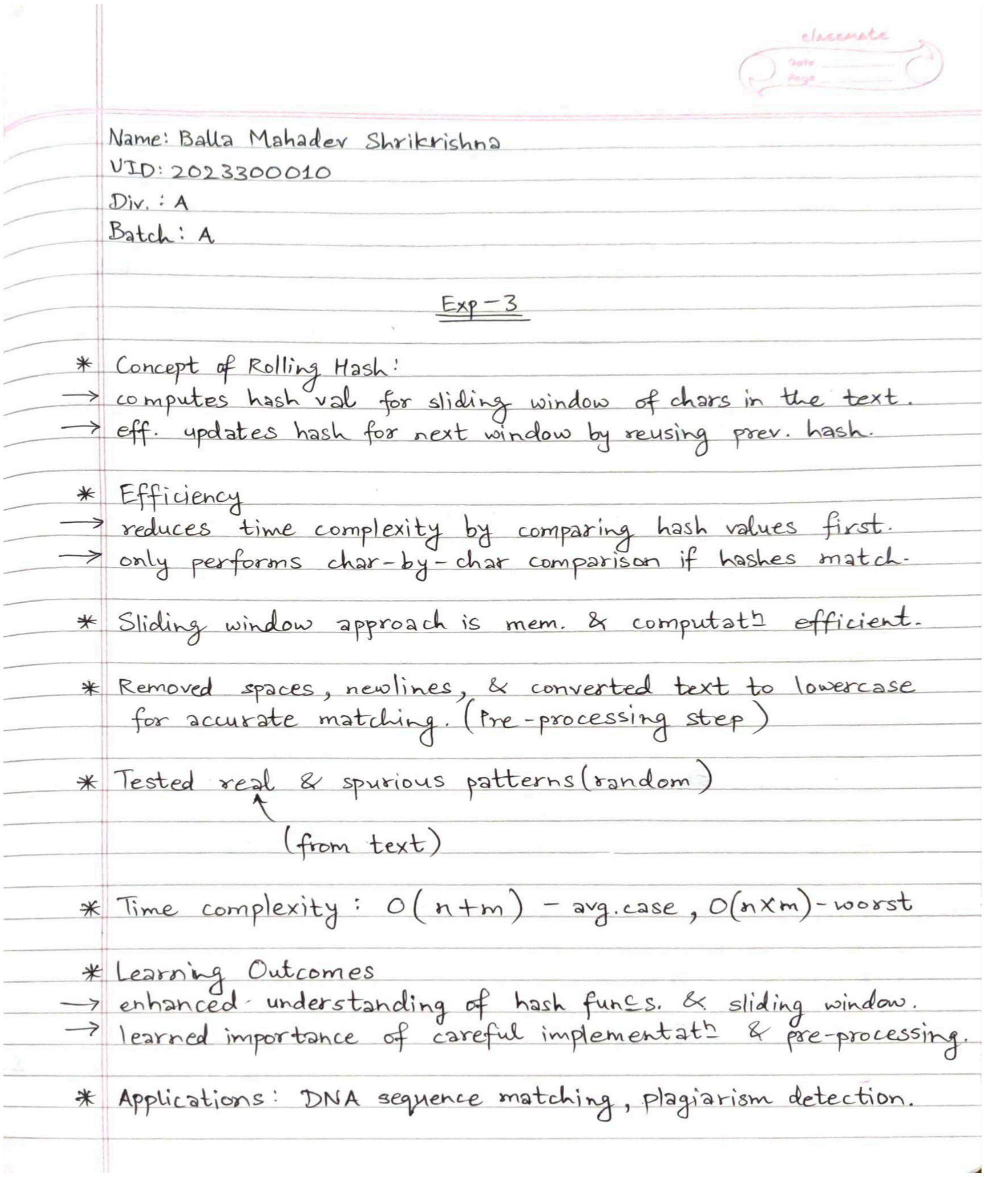
```
● mahadev@mahadev-Inspiron-15-3520:~/Desktop/Mahadev/SE/Sem4/DAA/Lab/Lab Sessions/exp3$ g++ rabinkarp.cpp
● mahadev@mahadev-Inspiron-15-3520:~/Desktop/Mahadev/SE/Sem4/DAA/Lab/Lab Sessions/exp3$ ./a.out
  Pattern (10 chars) in input_1000.txt found 5 times, Time: 0.012 ms
  Pattern (20 chars) in input_1000.txt found 1 times, Time: 0.01 ms
  Pattern (30 chars) in input_1000.txt found 0 times, Time: 0.011 ms
  Pattern (40 chars) in input_1000.txt found 0 times, Time: 0.01 ms
  Pattern (50 chars) in input_1000.txt found 0 times, Time: 0.008 ms
  Pattern (60 chars) in input_1000.txt found 0 times, Time: 0.008 ms
  Pattern (70 chars) in input_1000.txt found 1 times, Time: 0.008 ms
  Pattern (80 chars) in input_1000.txt found 1 times, Time: 0.008 ms
  Pattern (90 chars) in input_1000.txt found 1 times, Time: 0.008 ms
  Pattern (100 chars) in input_1000.txt found 0 times, Time: 0.011 ms
  Pattern (10 chars) in input_2000.txt found 0 times, Time: 0.016 ms
  Pattern (20 chars) in input_2000.txt found 0 times, Time: 0.02 ms
  Pattern (30 chars) in input_2000.txt found 1 times, Time: 0.028 ms
  Pattern (40 chars) in input_2000.txt found 0 times, Time: 0.019 ms
  Pattern (50 chars) in input_2000.txt found 0 times, Time: 0.015 ms
  Pattern (60 chars) in input_2000.txt found 0 times, Time: 0.016 ms
  Pattern (70 chars) in input_2000.txt found 0 times, Time: 0.016 ms
  Pattern (80 chars) in input_2000.txt found 1 times, Time: 0.016 ms
  Pattern (90 chars) in input_2000.txt found 1 times, Time: 0.016 ms
  Pattern (100 chars) in input_2000.txt found 0 times, Time: 0.017 ms
  Pattern (10 chars) in input_3000.txt found 1 times, Time: 0.021 ms
  Pattern (20 chars) in input_3000.txt found 0 times, Time: 0.024 ms
  Pattern (30 chars) in input_3000.txt found 0 times, Time: 0.025 ms
  Pattern (40 chars) in input_3000.txt found 1 times, Time: 0.018 ms
  Pattern (50 chars) in input_3000.txt found 1 times, Time: 0.021 ms
  Pattern (60 chars) in input_3000.txt found 0 times, Time: 0.024 ms
  Pattern (70 chars) in input_3000.txt found 0 times, Time: 0.023 ms
  Pattern (80 chars) in input_3000.txt found 0 times, Time: 0.025 ms
  Pattern (90 chars) in input_3000.txt found 0 times, Time: 0.023 ms
  Pattern (100 chars) in input_3000.txt found 1 times, Time: 0.023 ms
  Pattern (10 chars) in input_4000.txt found 0 times, Time: 0.024 ms
  Pattern (20 chars) in input_4000.txt found 1 times, Time: 0.031 ms
  Pattern (30 chars) in input_4000.txt found 0 times, Time: 0.033 ms
  Pattern (40 chars) in input_4000.txt found 1 times, Time: 0.033 ms
  Pattern (50 chars) in input_4000.txt found 1 times, Time: 0.032 ms
  Pattern (60 chars) in input_4000.txt found 1 times, Time: 0.031 ms
  Pattern (70 chars) in input_4000.txt found 0 times, Time: 0.033 ms
  Pattern (80 chars) in input_4000.txt found 0 times, Time: 0.031 ms
  Pattern (90 chars) in input_4000.txt found 1 times, Time: 0.032 ms
  Pattern (100 chars) in input_4000.txt found 0 times, Time: 0.039 ms
```

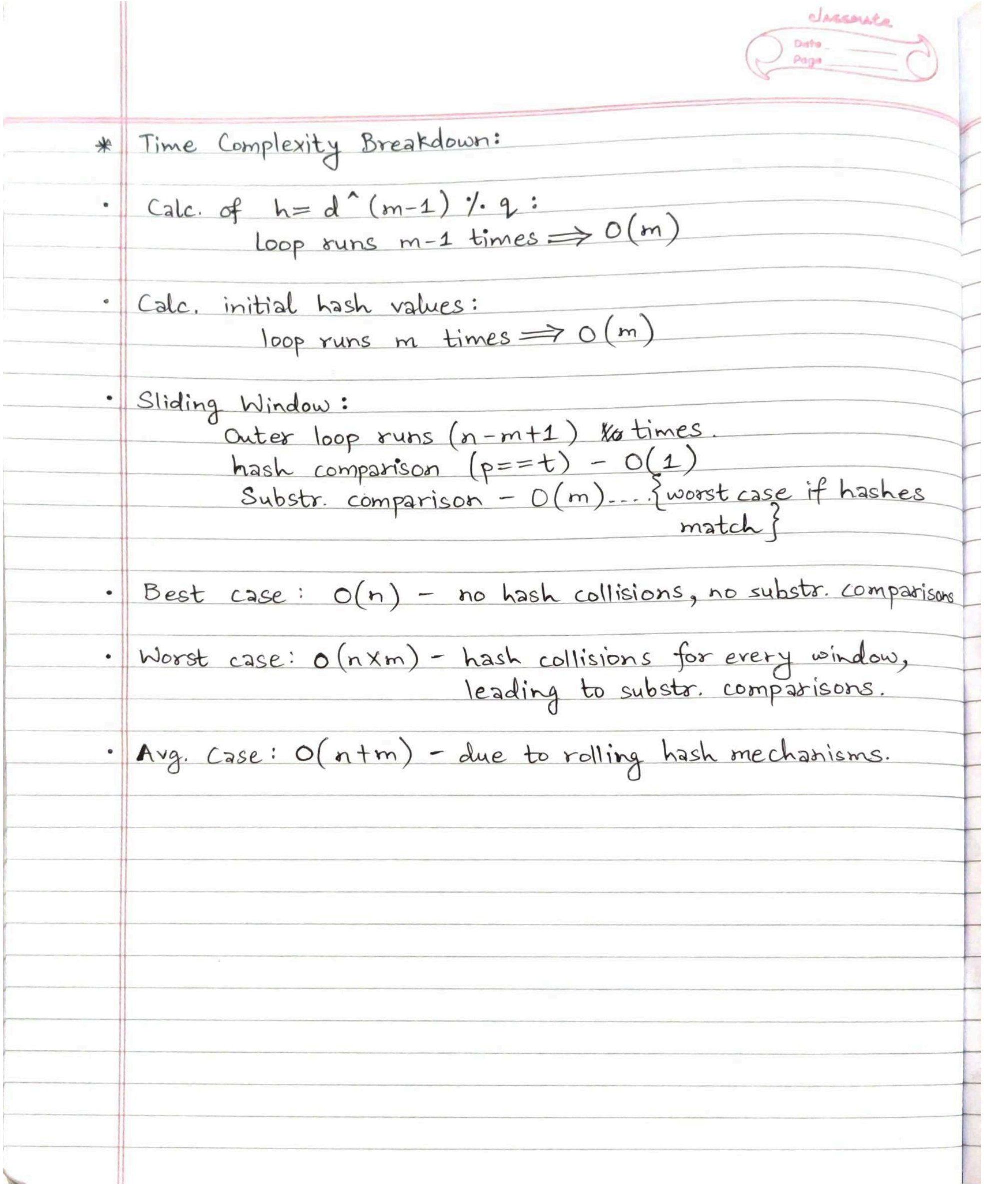
OUTPUT:



A	B	C	D	E	F
PatternSize	InputFile	Text(Ts)	Pattern(Ps)	TimeTaken(ms)	
10	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	rojectgute	0.012	
20	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	tingthisoranyotherpr	0.01	
30	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	YHQQOABJCETOCAEJPXGDJNJXZPOSRM	0.011	
40	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	QVEIKELVGQQWSRAEGYKJJZUJYJXRBLETGIBSMNPV	0.01	
50	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	GTXXUDFUQODPIMQTLJUWNPFWTZKIPHNXBKWXQBRGRWXBLNUWWR	0.008	
60	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	KGCIZBUHSCUPFHNCZQVFHREIESFBOWYAEAIGBDNUHKLMTZOUPLZZCFJJXOKM	0.008	
70	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	used.youcanalsofindoutabouthowtomakeadonationtoprojectgutenberg,andhow	0.008	
80	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	ojectgutenbergebook.thisheadershouldbethefirstthingseenwhenviewingthisprojectgut	0.008	
90	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	ectgutenbergebookoftheadventuresofsherlockholmesbysirarthurconandoyle(#15inourserie	0.008	
100	input_1000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	NCKJJWVVRWLLNYMMASWJPMVDXHSORPLFTXOCWLZNJMBXKNLMHJYZWVCVDW	0.011	
10	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	PHWBVGCUCL	0.016	
20	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	WTKNIOIRDZPQRUBEKFWL	0.02	
30	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	ecarbuncleviii.theadventureoft	0.028	
40	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	XARZUTLARGMHOBRHEQXXJTYPFFNSBSEYSVXNPKPI	0.019	
50	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	BPHEGRLYQIKJGZRLNJMHNMALLPCWEKRHCYOKRZJHHVTPUKAJVO	0.015	
60	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	LCTWNIYLOKCXMDLXUKICTFVIZHKLEAEPCZNQJLDXYGWKJIJDUTHQYEYANLNS	0.016	
70	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	RHQSVIBILAGTWTCGCIBYRTWYRYLELDSENKWKUXUHXDBWWFCAPFAHYZHSXUY	0.016	
80	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	theadventuresofsherlockholmesbysirarthurconandoyle(#15inourseriesbysirarthurcona	0.016	
90	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	eforedownloadingorredistributingthisoranyotherprojectgutenbergebook.thisheadershouldbe	0.016	
100	input_2000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	WMLZCTZLRASLYMMJMCABHOCDAYJLYQUXDHWHAXSUYMHYATJPYJSFYVKAVUK	0.017	
10	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	ecarbuncle	0.021	
20	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	VVHAJGWVQKOZWUJCRHED	0.024	
30	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	BHIOHUPXVIUTRWCZUARFQQDKCHCJOH	0.025	
40	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	preparedbythousandsofvolunteers!*****tit	0.018	
50	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	hewouldhaveplacedhimselfinafalseposition.heneversp	0.021	
60	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	VZUVXEPQXMUWGUNNLGSVIAZTOJNFXSIVUESTIJLFXFDEAQTNWOIGOKZETPJS	0.024	
70	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	TNDXHWFRJMOPPUPHQEGEMMSWOYPDHKMCZSZHQGYAUOPJJGQZKYDYLXXBX	0.023	
80	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	JTWUEEMDQLIQWHNZEEFKCXUCMDHRSOYDKWXOCKUUVEMTMCSSGXFIWZLKD	0.025	
90	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	XGQXRCCOGXZJYWLZFQJNSRXGRXWMPSFNYXMSAOIINHRODENKXYZPPYVJVTV	0.023	
100	input_3000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	lseposition.heneverspokeofthesofterpassions,savewithagibeandasneer.theywereadmirable	0.023	
10	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	WMXMQDQDPK	0.024	
20	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	rybeforedownloadingo	0.031	
30	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	ZEONNJZLTLGPSUFNVCLIZYACQFTAFO	0.033	
40	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	dez)theadventuresofsherlockholmesbysirar	0.033	
50	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	mysterieswhichhadbeenabandonedashopelessbytheoffic	0.032	
60	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	*****title:theadventuresofsherlockholmesauthor:sirarthurcona	0.031	
70	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	VVMGEBBYJQWLBGNBIRSNLSVBMEVUYFNWDAEHDFHOXDCBLPCTIUHTPEUDISY	0.033	
80	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	QWJRIYTBJQKCHOZKWRIFQWJTJUCCNVKDSTUAUPDDHOHOCGBZZLEQHPJTKNV	0.031	
90	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	okeofthesofterpassions,savewithagibeandasneer.theywereadmirablethingsfortheobserver-	0.032	
100	input_4000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	HTQAFEYEPFGSGGVDUDFRAPEZFRZWRFCYAUBGAZMPEVJNBGSXKYOMPVNWO	0.039	
10	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	entsi.asca	0.031	
20	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	renttohiscold,precis	0.032	
30	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	FNSYZHUCIEZWLLAKBXHBSWIQGWCBAD	0.038	
40	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	mofthisfile.includedisimportantinformati	0.032	
50	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	itingbyjosemenendez)theadventuresofsherlockholmesb	0.031	
60	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	QSBSBGRZRCABGZIHTGPJMNNPPQIWJTEZNFTOLMQCRQFXRQGNWYWLNMADEIX	0.037	
70	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	DORJHHWWXZNPEMJWSWTQUEEIGJNRIAUORNZAXVWUWJMBVVXQTSIPYMYFXL	0.036	
80	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	PNNSKKRAFKJVKXTPXENFTIBAQYFVQKWHXMZKWRMCDXZNXSFUYSZRCCRSAYP	0.036	
90	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	.ihadseenlittleofholmeslately.mymarriagehaddriftedusawayfromeachother.myowncomplete	0.037	
100	input_5000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	RAELCDBXOPZYPCTPXASYGTNKNJJGYITRIYCNDDKTUJRLLKBKMTJVMYHBISIIABZ	0.041	
10	input_6000.txt	theprojectgutenbergebookoftheadventuresofsherlockholmes	mitsuchint	0.036	


Pattern Size vs Time Taken for Different Input Files
Time Taken (ms)
Pattern Size (Characters)
0.05
0.10
0.15
0.20
0.25
20
40
60
80
100
Input File
input_1000.txt
input_2000.txt
input_3000.txt
input_4000.txt
input_5000.txt
input_6000.txt
input_7000.txt
input_8000.txt
input_9000.txt
input_10000.txt

CONCLUSION:	 Name: Balla Mahadev Shrikrishna UID: 2023300010 Div. : A Batch : A Exp - 3 * Concept of Rolling Hash: → computes hash val for sliding window of chars in the text. → eff. updates hash for next window by reusing prev. hash. * Efficiency → reduces time complexity by comparing hash values first. → only performs char-by-char comparison if hashes match. * Sliding window approach is mem. & computat^{n} efficient. * Removed spaces, newlines, & converted text to lowercase for accurate matching. (Pre-processing step) * Tested real & spurious patterns (random) ↑ (from text) * Time complexity : O(n+m) - avg.case, O(n×m) - worst * Learning Outcomes → enhanced understanding of hash func^{s}. & sliding window. → learned importance of careful implementat^{n} & pre-processing. * Applications: DNA sequence matching, plagiarism detection.



Date
Page

* Time Complexity Breakdown:

- Calc. of $h = d^\wedge (m-1) \% q$:
  loop runs $m-1$ times $\Longrightarrow O(m)$

- Calc. initial hash values:
  loop runs m times $\Longrightarrow O(m)$

- Sliding Window:
  Outer loop runs $(n-m+1)$ ~~to~~ times.
  hash comparison $(p==t)$ – $O(1)$
  Substr. comparison – $O(m)$... {worst case if hashes match}

- Best case: $O(n)$ – no hash collisions, no substr. comparisons

- Worst case: $O(n \times m)$ – hash collisions for every window, leading to substr. comparisons.

- Avg. Case: $O(n+m)$ – due to rolling hash mechanisms.