1-)

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

#define d 256

void search(char pat[], char txt[], int q) {

int M = strlen(pat);

int N = strlen(txt);

int i, j;

int p = 0;

int t = 0;

int h = 1;

for (i = 0; i < M - 1; i++)

h = (h \* d) % q;

for (i = 0; i < M; i++) {

p = (d \* p + pat[i]) % q;

t = (d \* t + txt[i]) % q;

}

for (i = 0; i <= N - M; i++) {

if (p == t) {

for (j = 0; j < M; j++) {

if (txt[i + j] != pat[j])

break;

}

if (j == M)

cout << "Pattern found at index " << i << endl;

}

if (i < N - M) {

t = (d \* (t - txt[i] \* h) + txt[i + M]) % q;

if (t < 0)

t = (t + q);

}

}

}

int main() {

char txt[] = "AABAACAADAABAAABAA";

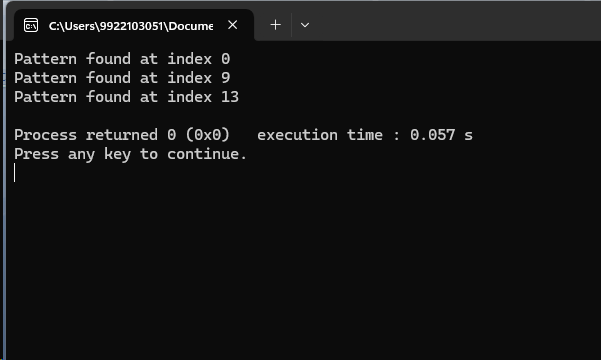
char pat[] = "AABA";

int q = 101;

search(pat, txt, q);

return 0;

}



2-)

#include <algorithm>

#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool WordBreak(const vector<string>& wordList, const string& word) {

int wordLen = word.length();

if (wordLen == 0) {

return true;

}

for (int i = 1; i <= wordLen; ++i) {

string prefix = word.substr(0, i);

if (find(wordList.begin(), wordList.end(), prefix) != wordList.end() &&

WordBreak(wordList, word.substr(i))) {

return true;

}

}

return false;

}

int main() {

vector<string> wordList = {

"mobile", "samsung", "sam", "sung", "man", "mango",

"icecream", "and", "go", "i", "like", "ice", "cream"

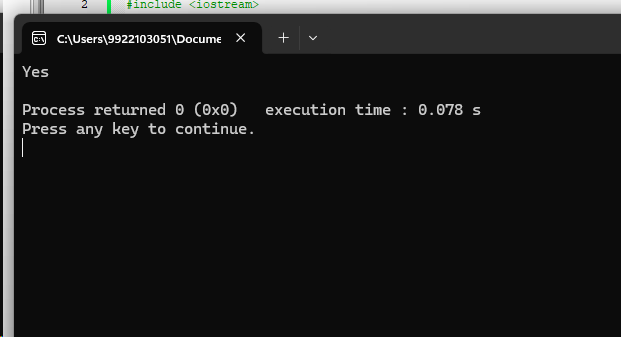
};

bool result = WordBreak(wordList, "ilikesamsung");

cout << (result ? "Yes" : "No") << endl;

return 0;

}



3-)

#include <iostream>

#include <set>

using namespace std;

void printUniqueRows(int matrix[][4], int rows, int cols) {

set<int> seen;

for (int i = 0; i < rows; ++i) {

int rowInt = 0;

for (int j = 0; j < cols; ++j) {

rowInt = (rowInt << 1) | matrix[i][j];

}

if (seen.count(rowInt) == 0) {

seen.insert(rowInt);

for (int j = 0; j < cols; ++j) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

}

int main() {

int matrix[3][4] = {

{1, 1, 0, 1},

{1, 0, 0, 1},

{1, 1, 0, 1}

};

int rows = sizeof(matrix) / sizeof(matrix[0]);

int cols = sizeof(matrix[0]) / sizeof(matrix[0][0]);

printUniqueRows(matrix, rows, cols);

return 0;

}

