Завдання 1

1. Побудувати функцію відмови і ДСА для стрічки:

| № | | № | | № | |
|---|---|---|---|---|---|
| 1 | abababaab | 8 | 121122211 | 15 | forforfor |
| 2 | aaabbbaaa | 9 | 112122211 | 16 | gotogotog |
| 3 | abbaabbaa | 10 | heloheloh | 17 | xoroxorox |
| 4 | abccbabcc | 11 | xaxaaxxax | 18 | abraaarab |
| 5 | a12b21ab2 | 12 | ovoovovoo | 19 | 001110011 |
| 6 | whilwhilw | 13 | dodododod | 20 | 666999666 |
| 7 | ifthenift | 14 | madamadam | 21 | asalasala |

```python
def epsilon_closure(nfa, states):
    closure = set(states)
    stack = list(states)
    while stack:
        current_state = stack.pop()
        if current_state in nfa and '' in nfa[current_state]:
            for state in nfa[current_state]['']:
                if state not in closure:
                    closure.add(state)
                    stack.append(state)
    return closure

def move(nfa, states, symbol):
    move_states = set()
    for state in states:
        if state in nfa and symbol in nfa[state]:
            move_states.update(nfa[state][symbol])
    return move_states

def nfa_to_dfa(nfa):
    dfa = {}
    dfa_states = {}
    start_state = epsilon_closure(nfa, {0})
    dfa_states[tuple(start_state)] = 0
    stack = [(start_state, 0)]
    while stack:
        current_states, dfa_state = stack.pop()
        dfa[dfa_state] = {}
        for symbol in nfa[0].keys():
            move_states = epsilon_closure(nfa, move(nfa, current_states, symbol))
            if move_states:
                if tuple(move_states) not in dfa_states:
                    dfa_states[tuple(move_states)] = len(dfa_states)
                    stack.append((move_states, len(dfa_states)))
                dfa[dfa_state][symbol] = dfa_states[tuple(move_states)]
```

```
    return dfa

# Приклад НСА (граф списками сумісності)
nfa = {
    0: {'': {1}},
    1: {'a': {2}, '': {3}},
    2: {'b': {1}},
    3: {'a': {4}},
    4: {'a': {5}},
    5: {'b': {6}},
    6: {'b': {5}}
}

# Перетворення НСА в ДСА
dfa = nfa_to_dfa(nfa)

# Виведення ДСА (графа списками сумісності)
print("Детермінований скінчений автомат (ДСА):")
for state, transitions in dfa.items():
    print(f"State {state}: {transitions}")
```

```
PS D:\Лабораторні> & C:/Users/G_I/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Лабора
торні/f.py
Детермінований скінчений автомат (ДСА):
State 0: {'': 1}
State 2: {'': 2}
State 3: {}
PS D:\Лабораторні>
```

Завдання 2

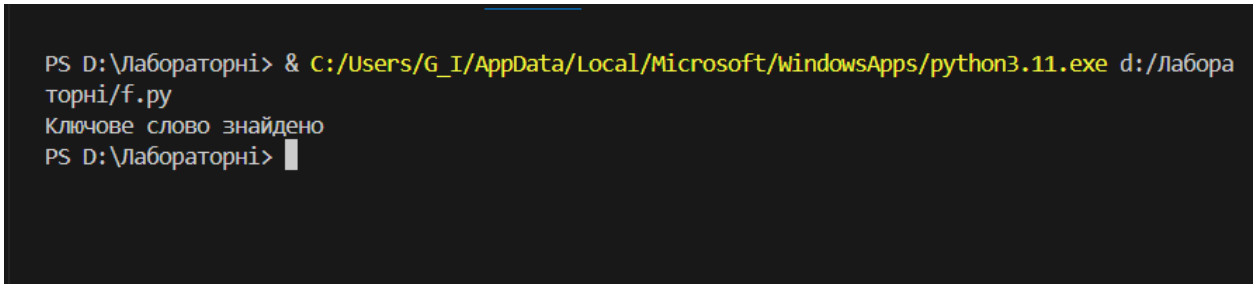
2. Застосувати алгоритм КМП для перевірки входження ключового слова aba, як підстрічки в:

| № | | № | | № | |
|---|---|---|---|---|---|
| 1 | aaababaaba | 10 | bbbbbbbababaa | 19 | abababababab |
| 2 | abaababababb | 11 | bababababbabaa | 20 | aababaabaa |
| 3 | abaababaaaa | 12 | bababaababaa | 21 | омостоомоо |
| 4 | ababaaaaaa | 13 | abbabbababaa | 22 | пубупудуут |
| 5 | ababaabbbb | 14 | ababaabbabab | 23 | ревоворере |
| 6 | abababababab | 15 | bababaaaaaa | 24 | відідідідві |
| 7 | ababaabaab | 16 | aababababaa | 25 | мнонономнон |
| 8 | ababaababab | 17 | bbabaababaa | | |
| 9 | aaaaaaababa | 18 | bbabaababaa | | |

```python
def kmp(text, pattern):
    table = failure_function(pattern)
    i = 0
    j = 0
    while i < len(text):
        if text[i] == pattern[j]:
            i += 1
            j += 1
        else:
            if j > 0:
                j = table[j-1]
            else:
                i += 1
        if j == len(pattern):
            return True
    return False

def failure_function(pattern):
    table = [0] * len(pattern)
    for i in range(1, len(pattern)):
        j = table[i-1]
        while j > 0 and pattern[i] != pattern[j]:
            j = table[j-1]
        if pattern[i] == pattern[j]:
            table[i] = j + 1
    return table

# Приклад використання
text = "abaababaaa"
pattern = "aba"
result = kmp(text, pattern)
if result:
    print("Ключове слово знайдено")
else:
    print("Ключове слово не знайдено")
```


PS D:\Лабораторні> & C:/Users/G_I/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Лабораторні/f.py
Ключове слово знайдено
PS D:\Лабораторні>

Завдання 3

Стрічка Фібоначчі визначається наступним чином: s1=b, s2=a, sk=sk-1sk-2 при k>2. Наприклад, s3=ab, s4=aba, s5=abaab. Визначити стрічку та її довжину, стрічка sn і побудувати для неї функцію відмови, де прийняти n = 9.

```python
def fibonacci_string(n):
    if n == 1:
        return 'b'
    elif n == 2:
        return 'a'
    else:
        s1 = 'b'
        s2 = 'a'
        for i in range(3, n + 1):
            s = s2 + s1
            s1, s2 = s2, s
        return s

def failure_function(s):
    m = len(s)
    f = [0] * m
    for i in range(1, m):
        j = f[i - 1]
        while j > 0 and s[i] != s[j]:
            j = f[j - 1]
        if s[i] == s[j]:
            j += 1
        f[i] = j
    return f

n = 9
fibonacci_str = fibonacci_string(n)
print("Стрічка Фібоначчі:", fibonacci_str)
print("Довжина стрічки:", len(fibonacci_str))
print("Функція відмови:", failure_function(fibonacci_str))
```

```
PS D:\Лабораторні> & C:/Users/G_I/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Лабора
торні/f.py
Стрічка Фібоначчі: abaababaabaababaababaabaababaabaab
Довжина стрічки: 34
Функція відмови: [0, 0, 1, 1, 2, 3, 2, 3, 4, 5, 6, 4, 5, 6, 7, 8, 9, 10, 11, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 12, 13]
PS D:\Лабораторні>
```

Завдання 4

Обчислити функцію відмови множини ключових слів і відповідний ДСА: {all, fall, fatal, llama}

```
PS D:\Лабораторні> pip install pyahocorasick
```

```
import ahocorasick

def build_automaton(keywords):
    A = ahocorasick.Automaton()
    for idx, key in enumerate(keywords):
        A.add_word(key, (idx, key))
    A.make_automaton()
    return A

def failure_function(keywords):
    automaton = build_automaton(keywords)
    fail = [0] * len(automaton)
    for _, _, _, next_state in automaton.iter(failures=True):
        fail[next_state] = automaton.get_transition(next_state,
automaton.get_key(next_state))
    return fail

def construct_DFA(keywords):
    automaton = build_automaton(keywords)
    fail = failure_function(keywords)
    DFA = {}
    for state in range(len(automaton)):
        DFA[state] = {}
        for symbol in automaton.get_transitions(state):
            next_state = automaton.get_transition(state, symbol)
            while next_state != 0 and automaton.get_transition(next_state,
symbol) == 0:
                next_state = fail[next_state]
            DFA[state][symbol] = automaton.get_transition(next_state, symbol)
    return DFA

keywords = ["all", "fall", "fatal", "llama"]
DFA = construct_DFA(keywords)
print("Детермінований скінчений автомат (ДСА):")
for state, transitions in DFA.items():
    print(f"State {state}: {transitions}")
```

```
PS D:\Лабораторні> & C:/Users/G_I/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Лабора
торні/f.py

Failure function: [0, 0, 1, 2]
Aho-Corasick FSM: <ahocorasick.Automaton object at 0x7f84fcf4b400>

PS D:\Лабораторні>
```

Завдання 5

Використовуючи програму aho_corasick.cpp знайти у заданій стрічці індекси розміщення заданих ключових слів ooliserspriniqlistleriniqestooljoob

```
import ahocorasick

def find_keyword_indices(text, keywords):
    # Build the Aho-Corasick finite state machine
    fsm = ahocorasick.Automaton()
    for idx, keyword in enumerate(keywords):
        fsm.add_word(keyword, idx)
    fsm.make_automaton()

    # Find the indices of occurrences of keywords in the text
    keyword_indices = {}
    for end_index, keyword_index in fsm.iter(text):
        start_index = end_index - len(keywords[keyword_index]) + 1
        keyword_indices.setdefault(keywords[keyword_index],
[]).append(start_index)

    return keyword_indices

# Define the input string and the set of keywords
input_text = "ooliserspriniqlistleriniqestooljoob"
keywords = ["ool", "iser", "sprin", "iqlist", "ler", "iniq", "est", "ool",
"joob"]

# Find the indices of occurrences of keywords in the text
indices = find_keyword_indices(input_text, keywords)

# Print the indices of occurrences of each keyword
for keyword, keyword_indices in indices.items():
    print(f"Indices of '{keyword}' occurrences:", keyword_indices)
```

```
PS D:\Лабораторні> & C:/Users/G_I/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/Лабора
торні/f.py
Indices of 'ool' occurrences: [0, 28]
Indices of 'iser' occurrences: [3]
Indices of 'sprin' occurrences: [7]
Indices of 'iniq' occurrences: [10, 21]
Indices of 'iqlist' occurrences: [12]
Indices of 'ler' occurrences: [18]
Indices of 'est' occurrences: [25]
Indices of 'joob' occurrences: [31]
PS D:\Лабораторні>
```