INDEX

Practical No :- 1. Write A Program For Tokenization Of Given Input.

Ans.

```
import re

def tokenize(input_string):
    tokens = re.findall(r'\w+|[^\w\s]', input_string, re.UNICODE)
    return tokens

if __name__ == "__main__":
    input_string = input("Enter a string to tokenize: ")
    tokens = tokenize(input_string)
    print("Tokens:", tokens)
```

OUTPUT :-

```
= RESTART: C:/Users/oes_lab/AppData/Local/Programs/Python/Python313/TOKENIZATION P1.py
Enter a string to tokenize: HELLO THIS IS GOOGLE GEMINI AI DEVELOPED UNDER GOOGLE DEEPMIND RESEARCH.
Tokens: ['HELLO', 'THIS', 'IS', 'GOOGLE', 'GEMINI', 'AI', 'DEVELOPED', 'UNDER', 'GOOGLE', 'DEEPMIND', 'RESEARCH', '.']
>>
```

Practical No :- 2. Write A Program For Generating Regular Expressions For Regular Grammar.

Ans.

```
import re

line = "horses are taller than dogs"
searchObj = re.search(r'(.*) are (.*?) .*', line, re.M|re.I)

if searchObj:
    print("searchObj.group() : ", searchObj.group())
    print("searchObj.group(1) : ", searchObj.group(1))
    print("searchObj.group(2) : ", searchObj.group(2))
else:
    print("Nothing found!!")
```

Output :-

```
>>
>>
========================================== RESTART: C:\Users\oes_lab\AppData\Local\
searchObj.group() :  horses are taller than dogs
searchObj.group(1) :  horses
searchObj.group(2) :  taller
>>
```

Practical No :- 3. Write A Program For Generating Derivation Sequence / Language For The Given Sequence Of Productions In Python.

Ans.

```
def printArray(arr, k):
    print(" ".join(map(str, arr)))

def getSuccessor(arr, k, n):
    for i in range(k - 1, -1, -1):
        if arr[i] < n:
            arr[i] += 1
            for j in range(i + 1, k):
                arr[j] = 1
            return 1
    return 0

def printSequences(n, k):
    arr = [0] * k
    for i in range(k):
        arr[i] = 1
    while True:
        printArray(arr, k)
        if getSuccessor(arr, k, n) == 0:
            break

n = 3
k = 2
printSequences(n, k)
```

Output :-

```
>>>
========== RESTART: D
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
>>>
```

Practical No :- 4. Design A Program For Creating A Machine That Accepts Three Consecutive One In Python.

Ans.

```
def stateA(n):
    if len(n) == 0:
        print("String Not Accepted")
    elif n[0] == '1':
        stateB(n[1:])
    else:
        stateA(n[1:])

def stateB(n):
    if len(n) == 0:
        print("String Not Accepted")
    elif n[0] == '1':
        stateC(n[1:])
    else:
        stateA(n[1:])

def stateC(n):
    if len(n) == 0:
        print("String Not Accepted")
    elif n[0] == '1':
        stateD(n[1:])
    else:
        stateA(n[1:])

def stateD(n):
    print("String Accepted")

input_string = "11101"  # Example input string
stateA(input_string)
```

Output :-

```
>>>
========== RESTART: D:/PY/PRAC4.py ==
String Accepted
>>>
```

Practical No :- 5. Design A Program For Creating Machine That Accepts The String Always Ending With 101 In Python

Ans.

```
def q1(s, i):
    print("q1->", end="")
    if i == len(s):
        print("NO")
        return
    if s[i] == '0':
        q1(s, i + 1)
    else:
        q3(s, i + 1)

def q2(s, i):
    print("q2->", end="")
    if i == len(s):
        print("NO")
        return
    if s[i] == '0':
        q1(s, i + 1)
    else:
        q3(s, i + 1)

def q3(s, i):
    print("q3->", end="")
    if i == len(s):
        print("YES")
        return
    if s[i] == '0':
        q1(s, i + 1)
    else:
        q2(s, i + 1)

def q0(s, i):
    print("q0->", end="")
    if i == len(s):
        print("NO")
        return
    if s[i] == '0':
```

```
        q1(s, i + 1)
    else:
        q2(s, i + 1)

if __name__ == "__main__":
    s = "010101"
    print("State transitions are", end=" ")
    q0(s, 0)
```

Output :-

```
============================== RESTART: D:/PRAC5.py =====
State transitions are q0->q1->q3->q1->q3->q1->q3->YES
```

Practical No :- 6. Design A Program For Accepting Decimal Number Divisible By 2 In Python.

Ans.

```
def stateq0(n):
    print(f"stateq0 called with {n}")
    if len(n) == 0:
        print("Input processed in stateq0")
    elif n[0] == '0':
        stateq0(n[1:])
    elif n[0] == '1':
        stateq1(n[1:])

def stateq1(n):
    print(f"stateq1 called with {n}")
    if len(n) == 0:
        print("Input processed in stateq1")
    elif n[0] == '0':
        stateq0(n[1:])
    elif n[0] == '1':
        stateq1(n[1:])

n = int(input("Enter a number: "))
n = bin(n).replace("0b", "")
stateq0(n)
```

Output :-

```
>>>
============================== RESTART: D:
Enter a number: 200
stateq0 called with 11001000
stateq1 called with 1001000
stateq1 called with 001000
stateq0 called with 01000
stateq0 called with 1000
stateq1 called with 000
stateq0 called with 00
stateq0 called with 0
stateq0 called with
Input processed in stateq0
>>>
```

Practical No :- 7. Design A Program For Creating A Machine Which Accepts String Having Equal No Of 1's And 0's In Python.

Ans.

```
# Function to count substrings with equal numbers of 0's, 1's,
and 2's
def getSubStringWithEqual012(Str):
    count = 0  # Counter for valid substrings

    # Traverse all possible substrings
    for i in range(len(Str)):
        countZero = 0
        countOnes = 0
        countTwo = 0

        # Check for every substring starting from index i
        for j in range(i, len(Str)):
            if Str[j] == '0':
                countZero += 1
            elif Str[j] == '1':
                countOnes += 1
            elif Str[j] == '2':
                countTwo += 1

            # If counts of 0's, 1's, and 2's are equal
            if countZero == countOnes and countOnes == countTwo:
                count += 1

    return count

# Driver code
Str = input("Enter the string: ")  # Take input from the user at
runtime
print("Number of substrings with equal 0's, 1's, and 2's:",
getSubStringWithEqual012(Str))
```

Output :-

```
>
============================== RESTART: D:/PY/PRAC7.py ===
Enter the string: 10021020
Number of substrings with equal 0's, 1's, and 2's: 3
>
```

Practical No :- 8. Design A Program For Creating A Machine Which Count The Number Of 1's And 0's In A Given String In Python.

Ans.

```
def countSubstring(S, n):
    ans = 0
    i = 0

    while i < n:
        cnt0 = 0
        cnt1 = 0

        # Count consecutive 0's
        while i < n and S[i] == '0':
            cnt0 += 1
            i += 1

        # Count consecutive 1's after consecutive 0's
        while i < n and S[i] == '1':
            cnt1 += 1
            i += 1

        # Update the total count of substrings with the minimum
of cnt0 and cnt1
        ans += min(cnt0, cnt1)

    return ans

# Driver code
if __name__ == "__main__":
    S = "0001110010"  # Input string
    n = len(S)        # Length of the string

    # Function call to count and print the result
    print("Total substrings:", countSubstring(S, n))
```

Output :-

```
============================== RESTART: D:/PY/PRAC8 ==
Total substrings: 4
```

Practical No 9. Design A PDA To Accept WCWR Here W Is Any String And Wr Is Reverse Of That String And C Is A Special Symbol.

Ans.

```
class DPDA:
    def __init__(self, trf, input, state):
        self.head = 0  # Tracks the position in the input string
        self.trf = trf  # Transition function
        self.state = state  # Initial state
        self.input = input  # Input string
        self.stack = ['Z']  # Initial stack with 'Z' as the
stack bottom marker

    def step(self):
        if self.head < len(self.input):
            a = self.input[self.head]
            s = self.stack.pop() if self.stack else 'ε'

            if (self.state, a, s) in self.trf:
                state, ss = self.trf.get((self.state, a, s))
                if ss != 'ε':  # Push symbols onto the stack in
reverse order
                    for symbol in reversed(ss):
                        self.stack.append(symbol)
                self.state = state
                print('{:20s} [{:10s}] {:5s}'.format(
                    self.input[self.head:], ''.join(self.stack),
self.state))
                self.head += 1
            else:
                print("No transition available!")
        else:
            print("Input fully processed!")

    def run(self):
        print('{:20s} [{:10s}] {:5s}'.format(
            self.input[self.head:], ''.join(self.stack),
self.state))
        while self.head < len(self.input):
            self.step()
```

```
        s = self.stack.pop() if self.stack else 'ε'
        if (self.state, 'ε', s) in self.trf:
            self.state, _ = self.trf.get((self.state, 'ε', s))
        print("Final state:", self.state)
        if self.state == 'q_accept':
            print("Accepted!")
        else:
            print("Rejected!")


# Transition function for DPDA accepting L = {a^n b^n | n >= 0}
trf = {
    ('q0', 'a', 'Z'): ('q0', 'AZ'),
    ('q0', 'a', 'A'): ('q0', 'AA'),
    ('q0', 'b', 'A'): ('q1', 'ε'),
    ('q1', 'b', 'A'): ('q1', 'ε'),
    ('q1', 'ε', 'Z'): ('q_accept', 'ε')  # Epsilon transition
for stack-empty check
}

# Driver code
if __name__ == "__main__":
    input_string = input("Enter a string (e.g., aaabbb): ")
    dpda = DPDA(trf, input_string, "q0")
    dpda.run()
```

Output :-

```
>>>
============================== RESTART: D:/PY/PRAC9.py =====
Enter a string (e.g., aaabbb): aaabbb
aaabbb                  [Z          ] q0
aaabbb                  [ZA         ] q0
aabbb                   [ZAA        ] q0
abbb                    [ZAAA       ] q0
bbb                     [ZAA        ] q1
bb                      [ZA         ] q1
b                       [Z          ] q1
Final state: q_accept
Accepted!
>>>
```

Practical No :- 10. Design A Turing Machine That's Accepts The Following Language b n c n Where N>0.

Ans.

```
# Function to perform actions for states
def action(inp, rep, move):
    global tapehead
    if tape[tapehead] == inp:
        tape[tapehead] = rep
        if move == 'L':  # Move the tape head left
            tapehead -= 1
        elif move == 'R':  # Move the tape head right
            tapehead += 1
        return True
    return False

# Initialize the tape
tape = ['B'] * 50  # 'B' represents blank spaces on the tape
string = input("Enter a string (e.g., aaabbbccc): ")
i = 5  # Start placing input at position 5 on the tape
tapehead = 5  # Initial position of the tape head

# Place the input string on the tape
for s in string:
    tape[i] = s
    i += 1

# Variables for the Turing Machine
state = 0
a, b, c, X, Z, U, V, R, L, B = 'a', 'b', 'c', 'X', 'Z', 'U',
'V', 'R', 'L', 'B'
oldtapehead = -1
accept = False

# Turing Machine logic
while oldtapehead != tapehead:  # If tape head does not move,
terminate
    oldtapehead = tapehead

    if state == 0:  # Initial state
```

```
        if action(a, X, R):  # Replace 'a' with 'X' and move
right
            state = 1
        elif action(B, B, R):  # Blank detected, transition to
final state
            state = 10
        elif action(Z, Z, R):  # Marker detected, move right
            state = 7
        elif action(b, U, R):  # Handle 'b', move right
            state = 4

    elif state == 1:
        if action(a, a, R):  # Skip over 'a'
            state = 1
        elif action(b, b, R):  # Transition to handle 'b'
            state = 2

    elif state == 2:
        if action(b, U, R):  # Replace 'b' with 'U' and move
right
            state = 3
        elif action(c, c, R):  # Skip over 'c'
            state = 5

    elif state == 3:
        if action(c, V, L):  # Replace 'c' with 'V' and move
left
            state = 4
        elif action(B, B, R):  # Transition to accept state if
all symbols processed
            state = 10

    elif state == 4:
        if action(b, b, L):  # Handle 'b' by moving left
            state = 4
        elif action(U, U, R):  # Skip 'U', transition back
            state = 5

    elif state == 5:
        if action(X, X, R):  # Skip over 'X', return to state 0
            state = 0
```

```
    elif state == 7:
        if action(Z, Z, R):  # Keep moving right over 'Z'
            state = 7
        elif action(b, U, R):  # Replace 'b' with 'U'
            state = 8

    elif state == 8:
        if action(c, V, R):  # Replace 'c' with 'V'
            state = 9
        elif action(B, B, R):  # Transition to accept state
            state = 10

    elif state == 10:  # Accept state
        accept = True
        break

# Output results
if accept:
    print("String accepted on state =", state)
else:
    print("String not accepted on state =", state)
```

Output :-

```
>>>
========== RESTART: D:/PY/PRAC10.py ==========
Enter a string (e.g., aaabbbccc): aaabbbccc
String not accepted on state = 3
>>>
========== RESTART: D:/PY/PRAC10.py ==========
Enter a string (e.g., aaabbbccc): aaabbb
String not accepted on state = 3
>>>
========== RESTART: D:/PY/PRAC10.py ==========
Enter a string (e.g., aaabbbccc): aaabbbcccddd
String not accepted on state = 3
>>>
```