CSA0888 – PYTHON PROGRAMMING

NAME: P.NANDINI

REG.NO: 192120020

DATE: 17 – 08 – 23

1 .You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?.

INPUT:

```
def climbStairs(n):
    if n <= 1:
        return 1
    fib = [0] * (n + 1)
    fib[1] = 1
    fib[2] = 2
    for i in range(3, n + 1):
        fib[i] = fib[i - 1] + fib[i - 2]
    return fib[n]
n = int(input("Enter the number of steps: "))
ways = climbStairs(n)
print("Number of distinct ways:", ways)
```

2.LEAP YEAR OR NOT

INPUT :

```
def is_leap_year(year):
    if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
        return True
    else:
        return False
```

```
year = int(input("Enter a year: "))
if is_leap_year(year):
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

3. MAXIMUM NUMBER OF WORDS FOUND IN SENTENCES

INPUT:

```
def max_words_in_sentence(sentences):
    max_words = 0
    for sentence in sentences:
        words = sentence.split()
        max_words = max(max_words, len(words))
    return max_words
sentences = ["Hello world", "This is a sentence", "Python programming"]
max_words = max_words_in_sentence(sentences)
print("Maximum number of words in a single sentence:", max_words)
```

4. MERGE TWO SORTED LISTS

INPUT:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def merge_sorted_lists(list1, list2):
    dummy_head = ListNode()
    current = dummy_head
    while list1 and list2:
        if list1.val < list2.val:
            current.next = list1
            list1 = list1.next
```

```
        else:
            current.next = list2
            list2 = list2.next
        current = current.next
    current.next = list1 if list1 else list2
    return dummy_head.next
list1 = ListNode(1, ListNode(3, ListNode(5)))
list2 = ListNode(2, ListNode(4, ListNode(6)))
merged_head = merge_sorted_lists(list1, list2)
while merged_head:
    print(merged_head.val, end=" -> ")
    merged_head = merged_head.next
```

5. . BASIC CALCULATOR

INPUT:

```
def calculate(s):
    stack = []
    num = 0
    operator = "+"
    operators = {"+", "-", "*", "/"}
    for i, char in enumerate(s):
        if char.isdigit():
            num = num * 10 + int(char)
        if char in operators or i == len(s) - 1:
            if operator == "+":
                stack.append(num)
            elif operator == "-":
                stack.append(-num)
            elif operator == "*":
                stack[-1] *= num
```

```
            elif operator == "/":
                stack[-1] = int(stack[-1] / num)
            operator = char
            num = 0
    return sum(stack)
expression = "3+2*2"
result = calculate(expression)
print("Result:", result)
```

6. GENERATE PARENTHESES

INPUT:

```
def generate_parentheses(n):
    def backtrack(s, left, right):
        if len(s) == 2 * n:
            result.append(s)
            return
        if left < n:
            backtrack(s + '(', left + 1, right)
        if right < left:
            backtrack(s + ')', left, right + 1)
    result = []
    backtrack("", 0, 0)
    return result
n = 1
combinations = generate_parentheses(n)
print(combinations)
```

7. THE MATCHING SHOULD COVER THE ENTIRE INPUT STRING

INPUT:

```
def is_match(s, p):
    m, n = len(s), len(p)
```

```
    dp = [[False] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = True
    for i in range(m + 1):
        for j in range(1, n + 1):
            if p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2] or (i > 0 and (s[i - 1] == p[j - 2] or p[j - 2] == '.') and dp[i - 1][j])
            else:
                dp[i][j] = i > 0 and (s[i - 1] == p[j - 1] or p[j - 1] == '.') and dp[i - 1][j - 1]
    return dp[m][n]
s = "mississippi"
p = "mis*is*p*."
result = is_match(s, p)
print("Is match:", result)
```

8. . THE YEAR IS DIVIDED INTO FOUR SEASONS

INPUT:

```
def get_season(month, day):
    if (month == "Dec" and day >= 21) or (month == "Jan" or month == "Feb") or (month == "Mar" and day < 20):
        return "Winter"
    elif (month == "Mar" and day >= 20) or (month == "Apr" or month == "May") or (month == "Jun" and day < 21):
        return "Spring"
    elif (month == "Jun" and day >= 21) or (month == "Jul" or month == "Aug") or (month == "Sep" and day < 22):
        return "Summer"
    else:
        return "Fall"
month = input("Enter the month (abbreviated, e.g., Jan, Feb, Mar, ...): ")
day = int(input("Enter the day of the month: "))
season = get_season(month, day)
```

```
print("The season for the date {} {} is: {}".format(month, day, season))
```

9. PYTHON PROGRAM TO REMOVE WORDS THAT ARE COMMON IN TWO STRINGS

INPUT:

```
def remove_common_words(s1, s2):
    words1 = set(s1.split())
    words2 = set(s2.split())
    common_words = words1.intersection(words2)
    unique_words1 = words1 - common_words
    unique_words2 = words2 - common_words
    result1 = ' '.join(unique_words1)
    result2 = ' '.join(unique_words2)
    return result1, result2
sentence1 = "sky is blue in color"
sentence2 = "Raj likes sky blue color"
result1, result2 = remove_common_words(sentence1, sentence2)
print("Output:")
print(result1)
print(result2)
```