COMP 1023 Introduction to Python Programming

Exercises on Recursion

COMP 1023 Teaching Team

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology, Hong Kong SAR, China

Exercise 1

Write a recursive function that computes the sum of the digits in an integer n. Use the following function header:

```
def sumDigits(n):
```

For example, `sumDigits(234)` returns $2 + 3 + 4 = 9$. Write a test program that prompts the user to enter an integer and displays the sum of its digits.

```
Enter an integer: 943210
The sum of digits in 943210 is 19
```

Exercise 1 Solutions

```python
def sumDigits(n):
    # Base case: if n is a single digit, return n
    if n < 10:
        return n
    # Recursive case: last digit + sum of remaining digits
    else:
        return n % 10 + sumDigits(n // 10)

# Test program
def main():
    number = int(input("Enter an integer: "))
    result = sumDigits(number)
    print(f"The sum of digits in {number} is {result}")

if __name__ == "__main__":
    main()
```

Exercise 2

Write a recursive function that counts down from a given number n to 0. Use the following function header:

```
def countdown(n):
```

Write a test program that prompts the user for a number and then prints a countdown in *'s from that number to 0 and then prints "Blast off!". For example, with an input of 4, the program should print

```
****
***
**
*
Blast off!
```

Exercise 2 Solutions

```python
def countdown(n):
    # Base case: when we reach 0
    if n == 0:
        print("Blast off!")
    # Recursive case: print stars and count down
    else:
        print('*' * n)
        countdown(n - 1)

# Test program
def main():
    number = int(input("Enter a number: "))
    countdown(number)

if __name__ == "__main__":
    main()
```

Exercise 3

Write a recursive function that calculates the result of $base^{exponent}$. Use the following function header:

```
def power(base, exponent):
```

Write a test program that prompts the user for a base and the exponent, calculates the result using the recursive function, and displays the result.

```
Enter a base: 40
Enter the exponent: 3
40 to the power of 3 is 64000
```

Exercise 3 Solutions

```python
def power(base, exponent):
    # Base case: any number to the power of 0 is 1
    if exponent == 0:
        return 1
    # Recursive case: base * base^(exponent-1)
    else:
        return base * power(base, exponent - 1)

# Test program
def main():
    base = int(input("Enter a base: "))
    exponent = int(input("Enter the exponent: "))
    result = power(base, exponent)
    print(f"{base} to the power of {exponent} is {result}")

if __name__ == "__main__":
    main()
```

Exercise 4

Write a recursive function to compute the following series:

$$m(i) = \frac{1}{3} + \frac{2}{4} + \ldots + \frac{i}{i+2}$$

Use the following function header:

```
def m(i):
```

Write a test program that prompts the user to enter an integer for `i` and displays `m(i)`.

```
Enter i: 15
m(15) is 11.120894954718484
```

Exercise 4 Solutions

```python
def m(i):
    # Base case: when i is 1, return 1/3
    if i == 1:
        return 1 / 3
    # Recursive case: current term + sum of previous terms
    else:
        return i / (i + 2) + m(i - 1)

# Test program
def main():
    i = int(input("Enter i: "))
    result = m(i)
    print(f"m({i}) is {result}")

if __name__ == "__main__":
    main()
```

Exercise 5

Write a recursive function that displays an integer value reversely on the console using the following header

```
def reverseDisplay(value):
```

Write a test program that prompts the user to enter an integer and displays its reversal.

```
Enter an integer: 12345
54321
```

Exercise 5 Solutions

```
def reverseDisplay(value):
    # Base case: single digit number
    if value < 10:
        print(value, end='')
    # Recursive case: display last digit, then reverse remaining digits
    else:
        print(value % 10, end='')
        reverseDisplay(value // 10)

# Test program
def main():
    number = int(input("Enter an integer: "))
    reverseDisplay(number)
    print()  # Add newline at the end

if __name__ == "__main__":
    main()
```

Exercise 6

Write a recursive function that returns the smallest number in a list *numbers*. Use the following function header:

```
def find_smallest(numbers):
```

Write a test program that prompts the user to enter series of numbers separated by spaces and displays the smallest one.

```
Enter numbers separated by spaces from one line: 123 5 654 22 12 78 21 54 -2 15 53

The smallest numbers in 123.0, 5.0, 654.0, 22.0, 12.0, 78.0, 21.0, 54.0, -2.0, 15.0,
53.0 is -2.0
```

Exercise 6 Solutions

```python
def find_smallest(numbers):
    # Base case: list with one element
    if len(numbers) == 1:
        return numbers[0]
    # Recursive case: compare first element with smallest of the rest
    else:
        smallest_of_rest = find_smallest(numbers[1:])
        return numbers[0] if numbers[0] < smallest_of_rest else smallest_of_rest

# Test program
def main():
    input_line = input("Enter numbers separated by spaces from one line: ")
    numbers = [float(x) for x in input_line.split()]

    if not numbers:
        print("No numbers entered!")
        return
    smallest = find_smallest(numbers)

    print("\nThe smallest numbers in ", end='')
    for i in range(len(numbers)):
        print(numbers[i], end='')
        if i < len(numbers) - 1:
            print(", ", end='')
    print(f" is {smallest}")

if __name__ == "__main__": main()
```

Exercise 7

Write a recursive function to find the greatest common divisor (GCD) of two positive integers *a* and *b* using Euclid's algorithm. Use the following function header:

```
def gcd(a, b):
```

Write a test program that prompts the user to enter two positive integers and displays their GCD.

The algorithm states:

- GCD(a, 0) = a
- GCD(a, b) = GCD(b, a mod b)

```
Enter the first positive integer: 48
Enter the second positive integer: 18
The GCD of 48 and 18 is 6
```

Exercise 7 Solutions

```python
def gcd(a, b):
    # Base case: when b becomes 0, a is the GCD
    if b == 0:
        return a
    # Recursive case: apply Euclidean algorithm
    # GCD(a, b) = GCD(b, a mod b)
    else:
        return gcd(b, a % b)


def main():
    num1 = int(input("Enter the first positive integer: "))
    num2 = int(input("Enter the second positive integer: "))

    result = gcd(num1, num2)

    print(f"The GCD of {num1} and {num2} is {result}")

if __name__ == "__main__":
    main()
```

Exercise 8

Write a recursive function that counts how many times a target value *target* appears in a list *lst*. Use the following function header:

```
def count_occurrences(lst, target):
```

Write a test program that prompts the user to enter a series of numbers separated by spaces and a target value to search for, then displays how many times the target appears in the list.

```
Enter a series of numbers separated by spaces: 2 5 2 8 2 9 2 1
Enter the target value to count: 2

The target value 2.0 appears 4 times in the list.
```

Exercise 8 Solutions

```python
def count_occurrences(lst, target):
    # Base case: if the list is empty, return 0 (no occurrences)
    if len(lst) == 0:
        return 0
    else:
        # Recursive case: count occurrences in the rest of the list
        count_rest = count_occurrences(lst[1:], target)

        if lst[0] == target:
            return 1 + count_rest
        else:
            return count_rest

def main():
    numbers_input = input("Enter a series of numbers separated by spaces: ")
    numbers = [float(x) for x in numbers_input.split()]
    target = float(input("Enter the target value to count: "))

    count = count_occurrences(numbers, target)

    print(f"\nThe target value {target} appears {count} times in the list.")

if __name__ == "__main__": main()
```