NAME: KEERTHANA M

USN: 1KI22CS056

COMPUTER SCIENCE AND ENGINEERING

KIT, TIPTUR

1 .Print the numbers in the pattern shown for input n. Sample output given when n=5

1 2 3 4 5

1 * 2 * 3 * 4 * 5

```
n=5
for i in range(1,6):
    print(i, end=' ')
print("\n")

for i in range(1, n+1):
    print(i, end=' ')
    if i != n:
        print('*', end=' ')
```

2. Print the numbers in the following pattern for input n. sample output given when n = 5

5 4 3 2 1

5 * 4 * 3 * 2 * 1

```
n=5
for i in range(n, 0, -1):
    print(i, end=' 'm)
print("\n")

for i in range(n, 1, -1):
    print(i, end=' * ')
print(1)
```

3. Print the numbers in pattern shown for input n. sample output given when n = 5

1 1 1 1 1

2 3 4 5 6

3 6 10 15 21

4 10 20 35 56

5 15 35 70 126

```
n = int(input("Enter a number: "))
def print_pattern(n):
    for i in range(1, n + 1):
        num = 0
        for j in range(1, n + 1):
            if i == 1:
                num = 1
            else:
                if j == 1:
                    num = i
                else:
                    num += (i - 1) * (j - 1)
            print(num, end=' ')
        print()

print_pattern(n)
```

4. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 *

2 **

3 ***

4 ****

5 *****

```
n=6
for i in range(1, n):
    print(i, end=' ')
    print('*' * i)
```

5. Print the numbers in the pattern shown for input n. sample output given when n = 5

5 *****

4 ****

3 ***

2 **

1 *

```
n = 6
for i in range(n-1, 0, -1):
    print(i, end=" ")
    print('*' *i)
```

6. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 ****

2 ***

3 **

4 *

5

```
n = 5
for i in range(1, n+1):
    print(i, '*'*(n-i))
```

7. Print the numbers in the pattern shown for input n. sample output given when n = 5

5 ****

4 ***

3 **

2 *

1

```
n = 5
for i in range(n, 0, -1):
    print(i, '*'*(i-1))
```

8. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 * 2 ** 3 *** 4 **** 5 *****

```
n = 5
for i in range(1, n+1):
    print(i, '*' *i, end=' ')
```

9. Print the numbers in the pattern shown for input n. sample output given when n = 5

5 ***** 4 **** 3 *** 2 ** 1 *

```
n = 5
for i in range(n, 0, -1):
    print(i, "*" *i, end =' ')
```

10. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 * 5

2 * 4

3 * 3

4 * 2

5 * 1

```
n = 5
for i in range(1, n+1):
```

```
    print(i, '*', n-i+1)
```

11. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 (0 stars)

2 ** (2 stars)

3 ****** (6 stars)

4 ********** (12 stars)

5 ********** (20 stars)

```
n = 5
for i in range(1, n+1):
    print(i, "*" *(i*(i-1)), f" {i*(i-1)} stars")
```

12. Print the numbers in the pattern shown for input n. sample output given when n = 5

1 * 5 * 5 * 25 * 125

2 * 4 * 8 * 32 * 256

3 * 3 * 9 * 27 * 243

4 * 2 * 8 * 16 * 128

5 * 1 * 5 * 05 * 025

```
n = 5
for i in range(1, n+1):
    val = i
    mult = n-i+1
    print(i, '*', mult, end =' ')
    for j in range(1, n+1):
        val *= mult
        print('*', val, end=' ')
    print()
```

13. Print a triangle of numbers for input n. sample output given when n = 6

01

02 03

04 05 06

07 08 09 10

11 12 13 14 15

16 17 18 19 20 21

```
n = 6
for i in range(1, n+1):
    for j in range(1, i+1):
        print(f"{(i*(i-1))//2 + j:02d}", end=' ')
    print()
```

14. Print an inverted triangle of numbers for input n. sample output given when n = 6

01 02 03 04 05 06

07 08 09 10 11

12 13 14 15

16 17 18

19 20

21

```
n= 6
for i in range(1, n+1):
    for j in range(1, n-i+2):
        print(f"{(i*(i-1))//2 + j:02d}", end=' ')
    print()
```

15. Print a triangle of numbers for input n. sample output given when n = 5

01

```
# 02 03
# 04 05 06
# 07 08 09 10
# 11 12 13 14 15
n = 5
for i in range(1, n+1):
    for j in range(1, i+1):
        print(f"{(i*(i-1))//2 +j:02d}", end=' ')
    print()
```

16. Print a spiral matrix of numbers for input n. sample output given when n = 5

```
01 * 02 * 03 * 04 * 05
06 * 07 * 08 * 09 * 10
11 * 12 * 13 * 14 * 15
16 * 17 * 18 * 19 * 20
21 * 22 * 23 * 24 * 25
```

```
m = 5
n = 5
num = 1
for i in range(m):
    for j in range(n):
        print(f"{num:02d}", end=' * ' if j<n-1 else ' ')
        num += 1
    print( )
```

17. Print a spiral matrix of numbers for input n. sample output given when n = 5

```
01 * 02 * 03 * 04 * 05
16 * 17 * 18 * 19 * 06
15 * 24 * 25 * 20 * 07
```

14 * 23 * 22 * 21 * 08

13 * 12 * 11 * 10 * 09

```
n = 5
matrix = [[0] * n for _ in range(n)]
num = 1
left, right = 0, n - 1
top, bottom = 0, n - 1
while left <= right and top <= bottom:
    for i in range(left, right + 1):
        matrix[top][i] = num
        num += 1
    top += 1
    for i in range(top, bottom + 1):
        matrix[i][right] = num
        num += 1
    right -= 1
    if top <= bottom:
        for i in range(right, left - 1, -1):
            matrix[bottom][i] = num
            num += 1
        bottom -= 1
    if left <= right:
        for i in range(bottom, top - 1, -1):
            matrix[i][left] = num
            num += 1
        left += 1
for row in matrix:
    for value in row:
        print(f"{value:02d}", end=' * ')
```

```
    print()
```

18. Print a spiral matrix of numbers for input n. sample output given when n = 5

```
# 01 * 02 * 03 * 04 * 05
# 10 * 09 * 08 * 07 * 06
# 11 * 12 * 13 * 14 * 15
# 20 * 19 * 18 * 17 * 16
# 21 * 22 * 23 * 24 * 25

n = 5
num = 1

for i in range(n):
    row = []
    for j in range(n):
        row.append(f"{num:02d}")
        num += 1

    if i % 2 == 1:
        row.reverse()

    print(" * ".join(row))
```

19. Print a spiral matrix of numbers for input n. sample output given when n = 5

```
 01*
  * 09 *
      * 13 *
          * 17 *
              * 25
```

```
def make_spiral(n):
    mat = [[0]*n for _ in range(n)]
    top, bottom, left, right = 0, n-1, 0, n-1
    num = 1
    while top <= bottom and left <= right:
        for j in range(left, right+1):
            mat[top][j] = num; num += 1
        top += 1
        for i in range(top, bottom+1):
            mat[i][right] = num; num += 1
        right -= 1
        if top <= bottom:
            for j in range(right, left-1, -1):
                mat[bottom][j] = num; num += 1
            bottom -= 1
        if left <= right:
            for i in range(bottom, top-1, -1):
                mat[i][left] = num; num += 1
            left += 1
    return mat

n = 5
mat = make_spiral(n)

# The exact positions that give your example order for n=5:
positions = [(0,0), (n-1,n-1), (n-1,0), (1,1), (2,2)]

for idx, (r, c) in enumerate(positions):
    val = mat[r][c]
    if idx == 0:
```

```
        print(f"{val:02d}*")
    else:
        indent = " " * (idx * 2)   # tweak multiplier to change indentation
        if idx == len(positions) - 1:
            print(f"{indent}* {val:02d}")
        else:
            print(f"{indent}* {val:02d} *")
```

20. Print the difference of two integers n and m without using the minus (-) operator.

```
m= int(input("Enter first integer :"))
n = int(input("Enter second integer :"))
count = 0
if n < m:
    while n != m:
        n += 1
        count += 1
    print(f"The difference of {m} and {n-count} is {count}")
elif n > m:
    while m != n:
        m += 1
        count += 1
    print(f"The difference of {m} and {n-count} is {count}")
else:
    print(f"The difference of {m} and {n} is 0")
```

21. Print the sum of two integers n and m without using the plus (+) operator.

```
m = int(input("Enter first integer: "))
n = int(input("Enter second integer: "))
```

```
m_original = m
for _ in range(n):
    m += 1
print(f"The sum of {m_original} and {n} is {m}")
```

22. Print the product of two integers n and m without using the multiplication (*) operator.

```
m = int(input("Enter first integer: "))
n = int(input("Enter second integer: "))

product = 0

negative = False
if m < 0 and n >= 0:
    m = -m
    negative = True
elif n < 0 and m >= 0:
    n = -n
    negative = True
elif m < 0 and n < 0:
    m = -m
    n = -n
for _ in range(n):
    product += m
if negative:
    product = -product
print(f"The product of {m} and {n} is {product}")
```

23. Print the division of two integers with numerator n and denominator m without using the division (/) operator and numerator is a multiple of denominator.

```
n = int(input("Enter numerator: "))
m = int(input("Enter denominator: "))

if m == 0:
    print("Division by zero is not allowed.")
else:
    quotient = 0
    temp = n

    while temp >= m:
        temp -= m
        quotient += 1

    print(f"{n} divided by {m} = {quotient}")
```

24. Given an integer, find the longest sequence of increasing digits. Sample inputs and outputs given:

Input: 9146826182 Output: 1468

Input: 924689128 Output: 24689

```
n = input("Enter an integer: ")
def longest_increasing_sequence(n):
    max_length = 0
    current_length = 1
    start_index = 0
    max_start_index = 0

    for i in range(1, len(n)):
        if n[i] > n[i - 1]:
            current_length += 1
        else:
```

```
            if current_length > max_length:
                max_length = current_length
                max_start_index = start_index
            current_length = 1
            start_index = i

    if current_length > max_length:
        max_length = current_length
        max_start_index = start_index

    return n[max_start_index:max_start_index + max_length]
result = longest_increasing_sequence(n)
print(f"The longest increasing sequence in {n} is: {result}")
```

25. Given an integer, find the longest sequence of increasing digits with incremental of 1. Sample inputs and outputs given

Input: 9146826182 Output: None

Input: 9145682618 Output: 456

```
n = input("Enter an integer: ")
def longest_increasing_sequence(n):
    max_length = 0
    current_length = 1
    start_index = 0
    max_start_index = 0

    for i in range(1, len(n)):
        if int(n[i]) == int(n[i - 1]) + 1:
            current_length += 1
        else:
            if current_length > max_length:
```

```
            max_length = current_length
            max_start_index = start_index
        current_length = 1
        start_index = i

    if current_length > max_length:
        max_length = current_length
        max_start_index = start_index

    return n[max_start_index:max_start_index + max_length] if max_length > 1 else None
result = longest_increasing_sequence(n)
if result:
    print(f"The longest increasing sequence in {n} is: {result}")
else:
    print(f"There is no increasing sequence in {n} with incremental of 1.")
```

26. Find a given digit n in a given integer m

```
# Input: n = 4, m = 124475577
# Output: 4

n = int(input("Enter the digit to find: "))
m = int(input("Enter the digit: "))
def find_digit(n, m):
    m_str = str(m)
    if str(n) in m_str:
        return n
    else:
        return "Digit not found"
result = find_digit(n, m)
print(result)
```

27. Print digits in a given integer as shown in the sample. The length of the number does not exceed 10 digits.

Input: 124475577

Output:

first digit : 1

second digit : 2

third digit : 4

fourth digit : 4

and so on

```
n = int(input("Enter an integer: "))
def print_digits(n):
    digits = str(n)
    for i, digit in enumerate(digits):
        print(f"{i + 1} digit: {digit}")
print_digits(n)
```

NOTE: enumerate(): is used in python to iterate over a sequence (like a list or string) to keep track of both index and value.

28. Print frequency of digits in a given integer. The length of the numberdoes not exceed 10 digits.

Input: 124475577

Output:

1 : 1 (first digit)

2 : 1 (second digit)

4 : 2 (third, fourth digits)

5 : 2 (sixth, seventh digits)

7 : 3 (fifth, eighth, ninth digits)

```
n = int(input("Enter an integer: "))
```

```
def print_digit_frequency(n):
    digits = str(n)
    frequency = {}

    for digit in digits:
        if digit in frequency:
            frequency[digit] += 1
        else:
            frequency[digit] = 1

    for digit, count in frequency.items():
        print(f"{digit} : {count} ({' '.join([f'{i + 1} digit' for i, d in enumerate(digits) if d == digit])})")

print_digit_frequency(n)
```

29. For input of odd integer n, print a half-rhombus shaped pattern using '*', example shown for n= 5

```
* * * * *
 * * * *
  * * *
   * *
    *
```

```
n = 5
for i in range(n):
    print(' ' * i + '* ' * (n - i))
```

30. For input of odd integer n, print a half-rhombus shaped pattern using '*', example shown for n = 7. Also print the mirror image of the output.

```
*
```

```
* *
* * *
* * * *
* * *
* *
*
```

```
n = 7
for i in range(n):
    if i < n // 2 + 1:
        print('* ' * (i + 1))
    else:
        print('* ' * (n - i))
```

31. For input of integer n, print a rhombus shaped pattern using '*', example shown for n = 4

```
   *
  * *
 * * *
* * * *
 * * *
  * *
   *
```

```
n = 4
# Top half
for i in range(1, n + 1):
    print(" " * (n - i) + "* " * i)
# Bottom half
for i in range(n - 1, 0, -1):
    print(" " * (n - i) + "* " * i)
```

32. Print the sums of all even and odd digits of an integer

Input: 12345678

Output:

Sum of all odd digits: 16

Sum of all even digits: 20

```
n = 12345678
def sum_even_odd_digits(n):
    odd_sum = 0
    even_sum = 0
    digits = str(n)

    for digit in digits:
        if int(digit) % 2 == 0:
            even_sum += int(digit)
        else:
            odd_sum += int(digit)

    print(f"Sum of all odd digits: {odd_sum}")
    print(f"Sum of all even digits: {even_sum}")
sum_even_odd_digits(n)
```

33. Print the sum digits of an integer m for a given condition

Input: m = 1132548 condition = odd

Output: 10

Input: m = 1132548 condition = even

Output: 14

```
m = 1132548
```

```
condition = 'odd'
def sum_digits(m, condition):
    total = 0
    digits = str(m)

    for digit in digits:
        if condition == 'odd' and int(digit) % 2 != 0:
            total += int(digit)
        elif condition == 'even' and int(digit) % 2 == 0:
            total += int(digit)

    return total
result = sum_digits(m, condition)
print(f"Sum of {condition} digits: {result}")
```

34. Replace each digit of a number with its word form

Input: 123

Output: one two three

```
n = int(input("Enter an integer: "))
def replace_digits_with_words(n):
    digit_to_word = {
        '0': 'zero',
        '1': 'one',
        '2': 'two',
        '3': 'three',
        '4': 'four',
        '5': 'five',
        '6': 'six',
        '7': 'seven',
```

```
        '8': 'eight',
        '9': 'nine'
    }

    words = [digit_to_word[digit] for digit in str(n)]
    return ' '.join(words)
result = replace_digits_with_words(n)
print(f"Number in words: {result}")
```

35. Print a 4-digit number in words form.

Input: 2345

Output: two thousand three hundred forty five

```
n = int(input("Enter a 4-digit integer: "))
def number_to_words(n):
    if n < 1000 or n > 9999:
        return "Input must be a 4-digit number."

    ones = {
        0: "",
        1: "one",
        2: "two",
        3: "three",
        4: "four",
        5: "five",
        6: "six",
        7: "seven",
        8: "eight",
        9: "nine"
    }
```

```
teens = {
    10: "ten",
    11: "eleven",
    12: "twelve",
    13: "thirteen",
    14: "fourteen",
    15: "fifteen",
    16: "sixteen",
    17: "seventeen",
    18: "eighteen",
    19: "nineteen"
}
tens = {
    2: "twenty",
    3: "thirty",
    4: "forty",
    5: "fifty",
    6: "sixty",
    7: "seventy",
    8: "eighty",
    9: "ninety"
}

words = []

# Thousands place
words.append(ones[n // 1000])
words.append("thousand")
```

```
    # Hundreds place
    if (n // 100) % 10 != 0:
        words.append(ones[(n // 100) % 10])
        words.append("hundred")

    # Tens and ones
    last_two = n % 100
    if last_two != 0:
        if last_two < 10:
            words.append(ones[last_two])
        elif last_two < 20:
            words.append(teens[last_two])
        else:
            words.append(tens[last_two // 10])
            if last_two % 10 != 0:
                words.append(ones[last_two % 10])

    return " ".join([w for w in words if w])  # Remove empty words

print("Number in words:", number_to_words(n))
```

36. For input n, print number of ways to form sum using 1s and 2s. Order matters.

Input: 4

Output: 5

1 1 1 1

1 1 2

1 2 1

2 1 1

2 2

```
n = 4
# Count ways (recursion)
def count_ways(n):
    if n < 0:
        return 0
    if n == 0:
        return 1
    return count_ways(n - 1) + count_ways(n - 2)

# Generate sequences (backtracking)
def generate_sequences(target, current):
    if sum(current) == target:
        print(" ".join(map(str, current)))
        return
    if sum(current) > target:
        return
    generate_sequences(target, current + [1])
    generate_sequences(target, current + [2])

ways = count_ways(n)
print(f"Number of ways to form sum {n} using 1s and 2s: {ways}")
generate_sequences(n, [])
```

37. Check if a number is a palindrome. without converting to string.

Input: 121 Output: 121 is a Palindrome

Input: 122 Output: 121 is not a Palindrome

```
n = int(input("Enter an integer: "))
def is_palindrome(n):
    original = n
```

```
    reversed_num = 0

    while n > 0:
        digit = n % 10
        reversed_num = reversed_num * 10 + digit
        n //= 10

    if original == reversed_num:
        print(f"{original} is a Palindrome")
    else:
        print(f"{original} is not a Palindrome")
is_palindrome(n)
```

38. Print a hollow square pattern of '*' for a given size n.

Input: n = 4

Output:

```
* * * *
*     *
*     *
* * * *
```

```
n = 4
def print_hollow_square(n):
    for i in range(n):
        for j in range(n):
            if i == 0 or i == n - 1 or j == 0 or j == n - 1:
                print("*", end=" ")
            else:
                print(" ", end=" ")
        print()
```

```
print_hollow_square(n)
```

39. Print a full square pattern of '*' for a given size n

Input: n = 4

Output:

```
* * * *
* * * *
* * * *
* * * *
```

```
n = 4
def print_full_square(n):
    for i in range(n):
        for j in range(n):
            print("*", end=" ")
        print()

print_full_square(n)
```

40. Write a program to print any alphabet using *

Input: A

Output:

```
   *
  * *
 *   *
*******
*     *
```

```
def print_A():
    rows = 5
```

```
    for i in range(rows):
        for j in range(rows * 2 - 1):
            if j == rows - i - 1 or j == rows + i - 1:  # sides
                print("*", end="")
            elif i == rows // 2 and j > rows - i - 1 and j < rows + i - 1:  # middle bar
                print("*", end="")
            else:
                print(" ", end="")
        print()
print_A()
```

41. Print a diamond pattern of numbers for a given odd number n

Input: 5

Output:

```
  1
 1 2 1
1 2 3 2 1
 1 2 1
  1
```

```
n = 5
def print_diamond_pattern(n):
    if n % 2 == 0:
        print("Please enter an odd number.")
        return

    mid = n // 2
    for i in range(n):
        if i <= mid:
            num = i + 1
```

```
        else:
            num = n - i

        spaces = ' ' * (mid - i) if i <= mid else ' ' * (i - mid)
        numbers = ' '.join(str(j) for j in range(1, num + 1))
        reverse_numbers = ' '.join(str(j) for j in range(num - 1, 0, -1))

        print(spaces + numbers + (' ' + reverse_numbers if num > 1 else ''))
print_diamond_pattern(n)
```

42. Write a program to reverse a number n Input: 1234 Output: 4321

```
n = int(input("Enter an integer: "))
rev = int(str(n)[::-1])
print(f"{n} will be {rev} after reversing.")
```

43. Write a program to add all even digits and multiply all odd digits of a given number n

Input: 1235

Output:

Sum of odd digits: 4

Sum of even digits: 7

```
n = int(input("Enter an integer: "))
def add_even_multiply_odd(n):
    even_sum = 0
    odd_product = 1
    has_odd = False  # To check if there are any odd digits

    digits = str(n)
```

```
    for digit in digits:
        if int(digit) % 2 == 0:
            even_sum += int(digit)
        else:
            odd_product *= int(digit)
            has_odd = True

    if not has_odd:  # If there are no odd digits, set product to 0
        odd_product = 0

    print(f"Sum of even digits: {even_sum}")
    print(f"Product of odd digits: {odd_product}")
add_even_multiply_odd(n)
```

44. Write a program to add all digits that are even and multiply all odd digits of a given number n

Input: 12350

Output:

Sum of digits that are even: 2

Product of digits that are odd: 15

```
n = int(input("Enter an integer: "))
def add_even_multiply_odd(n):
    even_sum = 0
    odd_product = 1
    has_odd = False

    for digit_char in str(n):
        digit = int(digit_char)
        if digit % 2 == 0:      # even
            even_sum += digit
```

```
        else:                   # odd
            odd_product *= digit
            has_odd = True

    # If no odd digits found, product should be 0
    if not has_odd:
        odd_product = 0

    print(f"Sum of digits that are even: {even_sum}")
    print(f"Product of digits that are odd: {odd_product}")

add_even_multiply_odd(n)
```

45. Write a program to display multiplication table for a given number n upto the multiplier m

Input: n = 10 m = 20

Output:

10 x 1 = 10

10 x 2 = 20

..

10 x 20 = 200

```
n = int(input("Enter the number for which you want the multiplication table: "))
m = int(input("Enter the multiplier limit: "))
def multiplication_table(n, m):
    for i in range(1, m + 1):
        print(f"{n} x {i} = {n * i}")
multiplication_table(n, m)
```

46. Write a program to accept N numbers and find the largest and smallest of them

Input: N = 10

n1 = 100 ; n2 = 202334; n3 = 3; ….

n10 = 6451

Output:

Smallest number is: 3

Largest number is: 202334

```
def find_largest_and_smallest(n):
    if n <= 0:
        print("Please enter a positive integer for N.")
        return

    numbers = []
    for i in range(n):
        num = int(input(f"Enter number {i + 1}: "))
        numbers.append(num)

    smallest = min(numbers)
    largest = max(numbers)

    print(f"Smallest number is: {smallest}")
    print(f"Largest number is: {largest}")
n = int(input("Enter the number of integers (N): "))
find_largest_and_smallest(n)
```

47. Write a program to search a digit n in a given number N

Input: N = 500; n = 0

Output: 0 occurs 2 times

```
N = int(input("Enter the number N: "))
n = int(input("Enter the digit n to search: "))
```

```
def search_digit(N, n):
    count = 0
    for digit in str(N):
        if int(digit) == n:
            count += 1
    return count
print(f"{n} occurs {search_digit(N, n)} times in {N}.")
search_digit(N, n)
```

48. Write a program find the nth digit of a number N

```
# Input: N = 50023; n = 4
# Output:4th digit is 2
```

```
N = int(input("Enter the number N: "))
n = int(input("Enter the digit n to find: "))
def find_nth_digit(N, n):
    digits = str(N)
    if n <= len(digits):
        return int(digits[n - 1])
    else:
        return None
result = find_nth_digit(N, n)
if result is not None:
    print(f"{n}th digit is {result}.")
else:
    print(f"{n}th digit does not exist in {N}.")
find_nth_digit(N, n)
```

49. Write a program that accepts 2 co-ordinates (x, y) and (x', y') of a rectangle of size n(length) x m(width) and finds the distance between them

```
import math
def distance_between_points(x1, y1, x2, y2):
    return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

# Input coordinates
x1, y1 = map(float, input("Enter the first coordinate (x1, y1): ").split())
x2, y2 = map(float, input("Enter the second coordinate (x2, y2): ").split())
# Calculate distance
distance = distance_between_points(x1, y1, x2, y2)
print(f"The distance between the points ({x1}, {y1}) and ({x2}, {y2}) is: {distance:.2f}")
```

50. Refer to Fresher Profiles – Student agreement and write a program to calculate the placement assistance fee a student would pay based on the CTC and placement drive type.

```
print("Placement Drive Types: Normal / Premium")
drive_type = input("Enter placement drive type: ").strip().lower()
ctc = float(input("Enter CTC (in LPA): "))

fee_percentage = 0

# Sample fee structure
if drive_type == "normal":
    if ctc <= 5:
        fee_percentage = 5
    elif ctc <= 10:
        fee_percentage = 7
    else:
        fee_percentage = 8

elif drive_type == "premium":
    if ctc <= 5:
```

```
        fee_percentage = 8
    elif ctc <= 10:
        fee_percentage = 10
    else:
        fee_percentage = 12
else:
    print("Invalid drive type entered.")
    exit()

fee_amount = (ctc * 100000 * fee_percentage) / 100  # converting LPA to actual ₹

print(f"\n--- Placement Assistance Fee Details ---")
print(f"CTC: ₹{ctc:.2f} LPA")
print(f"Drive Type: {drive_type.capitalize()}")
print(f"Fee Percentage: {fee_percentage}%")
print(f"Fee Amount: ₹{fee_amount:,.2f}")
```

51. A cricket batter is dismissed after facing n balls and scores m runs by hitting x fours, y sixes, z twos, and w ones. Write a program to calculate the strike rate, percentage contribution of each run type, and analyze data for p players to determine:

- The player with the highest strike rate
- The player who scored maximum runs
- The player who scored minimum runs
- The player who hit the most sixes
- The player who hit the most fours

```
p = int(input("Enter number of players: "))
players = []
for i in range(p):
    print(f"\nEnter data for Player {i+1}:")
    name = input("Name: ")
```

```
n = int(input("Balls faced: "))
m = int(input("Total runs: "))
x = int(input("Number of fours: "))
y = int(input("Number of sixes: "))
z = int(input("Number of twos: "))
w = int(input("Number of ones: "))

# Strike rate
strike_rate = (m / n) * 100 if n > 0 else 0

# Percent contribution of each run type
four_runs = x * 4
six_runs = y * 6
two_runs = z * 2
one_runs = w * 1

pct_fours = (four_runs / m * 100) if m > 0 else 0
pct_sixes = (six_runs / m * 100) if m > 0 else 0
pct_twos = (two_runs / m * 100) if m > 0 else 0
pct_ones = (one_runs / m * 100) if m > 0 else 0

players.append({
    "name": name,
    "runs": m,
    "balls": n,
    "fours": x,
    "sixes": y,
    "strike_rate": strike_rate,
    "pct_fours": pct_fours,
    "pct_sixes": pct_sixes,
```

```
        "pct_twos": pct_twos,
        "pct_ones": pct_ones
    })

# Display each player's stats
print("\nPlayer Statistics:")
for player in players:
    print(f"\nName: {player['name']}")
    print(f"Runs: {player['runs']} | Balls: {player['balls']} | Strike Rate: {player['strike_rate']:.2f}")
    print(f"Fours: {player['fours']} ({player['pct_fours']:.2f}%)")
    print(f"Sixes: {player['sixes']} ({player['pct_sixes']:.2f}%)")
    print(f"Twos: {player['pct_twos']:.2f}% | Ones: {player['pct_ones']:.2f}%")



# Analysis
highest_sr = max(players, key=lambda p: p['strike_rate'])
max_runs = max(players, key=lambda p: p['runs'])
min_runs = min(players, key=lambda p: p['runs'])
most_sixes = max(players, key=lambda p: p['sixes'])
most_fours = max(players, key=lambda p: p['fours'])

print("\n--- Analysis ---")
print(f"Highest Strike Rate: {highest_sr['name']} ({highest_sr['strike_rate']:.2f})")
print(f"Maximum Runs: {max_runs['name']} ({max_runs['runs']})")
print(f"Minimum Runs: {min_runs['name']} ({min_runs['runs']})")
print(f"Most Sixes: {most_sixes['name']} ({most_sixes['sixes']})")
print(f"Most Fours: {most_fours['name']} ({most_fours['fours']})")
```

52. Write a program to generate a receipt based on the items sold in a pizza shop. The format of the receipt is as below

---

Sr. No. Item Name Qty UoM Rate Sub-Total

-------+-----------------+------+------+-----------+-----------

1. + Cheesecake + 1.45 + Kg + 100.00 + 145.00

2. + Farmhouse Pizza + 3 + M + 1243.20 + 3729.80

3.

..

n. + Garlic Bread + 2 + Num + 149.99 + 299.98

-------+-----------------+------+------+-----------+-----------

TOTAL NNNNNN.NN

```
# Pizza Shop Receipt Generator

items = []
n = int(input("Enter number of items sold: "))

for i in range(n):
    print(f"\nEnter details for item {i+1}:")
    name = input("Item Name: ")
    qty = float(input("Quantity: "))
    uom = input("Unit of Measure (Kg/M/Num): ")
    rate = float(input("Rate: "))
    sub_total = qty * rate
    items.append((i+1, name, qty, uom, rate, sub_total))

# Print receipt
print("-" * 64)
print(f"{'Sr. No.':<7} {'Item Name':<18} {'Qty':>5} {'UoM':>6} {'Rate':>11} {'Sub-Total':>11}")
print("-" * 64)

total = 0
```

```
for sr, name, qty, uom, rate, sub_total in items:
    print(f"{sr:<7} {name:<18} {qty:>5.2f} {uom:>6} {rate:>11.2f} {sub_total:>11.2f}")
    total += sub_total

print("-" * 64)
print(f"{'TOTAL':>53} {total:>11.2f}")
print("-" * 64)
```

53. Write a program to print the area of a geometric shape such as a triangle, square, or circle etc. The program should prompt the user to specify the type of shape and then request the necessary dimensions to find its area

```
import math
print("Choose a shape to calculate area:")
print("1. Triangle")
print("2. Square")
print("3. Rectangle")
print("4. Circle")

choice = input("Enter shape name or number: ").strip().lower()

if choice in ("1", "triangle"):
    base = float(input("Enter base: "))
    height = float(input("Enter height: "))
    area = 0.5 * base * height
    print(f"Area of Triangle = {area:.2f}")

elif choice in ("2", "square"):
    side = float(input("Enter side length: "))
    area = side ** 2
    print(f"Area of Square = {area:.2f}")
```

```
elif choice in ("3", "rectangle"):
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    area = length * width
    print(f"Area of Rectangle = {area:.2f}")

elif choice in ("4", "circle"):
    radius = float(input("Enter radius: "))
    area = math.pi * radius ** 2
    print(f"Area of Circle = {area:.2f}")

else:
    print("Invalid shape selected!")
```

54. Write a program to find the length of a floating-point number. For eg., 1234.567 should result in length = 4,3

```
num = input("Enter a floating-point number: ")

if '.' in num:
    integer_part, decimal_part = num.split('.')
    int_len = len(integer_part)
    dec_len = len(decimal_part)
    print(f"Length = {int_len},{dec_len}")
else:
    print("Not a floating-point number!")
```