<u>WEB PROGRAMMING LAB</u>

LAB NO – 4   BASICS OF PYTHON

ADITYA AGARWAL   220905106   ROLL NO. 14

Q1 – Write a Python Program to select the smallest element from a list in expected linear time.

CODE:
```
import random
def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:  # We change this to <= to handle the smallest correctly
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1

def randomized_select(arr, low, high):
    if low == high:
        return arr[low]

    pivot_index = random.randint(low, high)  # Random pivot selection
    arr[pivot_index], arr[high] = arr[high], arr[pivot_index]  # Swap pivot with last element

    partition_index = partition(arr, low, high)
    # If partition index is at the first element, we return it (smallest element)
    if partition_index == 0:
        return arr[partition_index]
    else:
        return randomized_select(arr, low, partition_index - 1)

# Taking input from the user
input_list = input("Enter the list of numbers (comma separated): ").split(',')
arr = [int(num.strip()) for num in input_list]

# Finding the smallest element
smallest_element = randomized_select(arr, 0, len(arr) - 1)

# Display the result
print(f"The smallest element is: {smallest_element}")
```

OUTPUT:

```
 Enter the list of numbers (comma separated): 1, 5, 90, 22
The smallest element is: 1
```

Q2 - Implement the Bubble Sort algorithm to sort a list of numbers

CODE:

```
def bubble_sort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already sorted, no need to compare them
        for j in range(0, n-i-1):
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr

# Taking input from the user
input_list = input("Enter the list of numbers (comma separated): ").split(',')
arr = [int(num.strip()) for num in input_list]

# Sorting the array using bubble sort
sorted_array = bubble_sort(arr)

# Display the sorted array
print(f"Sorted array: {sorted_array}")
```

OUTPUT:

```
Enter the list of numbers (comma separated): 90, 32, 55, 12, 48
Sorted array: [12, 32, 48, 55, 90]
```

Q3 – Write a Python Programming to multiply two matrices

CODE:

```
# Function to multiply two matrices
def multiply_matrices(A, B):
    # Number of rows and columns in A
    rows_A = len(A)
    cols_A = len(A[0])

    # Number of rows and columns in B
    rows_B = len(B)
    cols_B = len(B[0])

    # Check if multiplication is possible (columns of A must equal rows of B)
    if cols_A != rows_B:
        raise ValueError("Matrix multiplication is not possible. The number of columns in A must be equal to the number of rows in B.")

    # Initialize the result matrix with zero values
    result = [[0 for _ in range(cols_B)] for _ in range(rows_A)]

    # Perform matrix multiplication
    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):  # or k in range(rows_B)
                result[i][j] += A[i][k] * B[k][j]

    return result
```

```
# Taking input from the user for matrix A
rows_A = int(input("Enter the number of rows for matrix A: "))
cols_A = int(input("Enter the number of columns for matrix A: "))
print("Enter the elements of matrix A row by row:")
A = []
for i in range(rows_A):
    row = list(map(int, input(f"Enter row {i + 1}: ").split()))
    A.append(row)

# Taking input from the user for matrix B
rows_B = int(input("Enter the number of rows for matrix B: "))
cols_B = int(input("Enter the number of columns for matrix B: "))
print("Enter the elements of matrix B row by row:")
B = []
for i in range(rows_B):
    row = list(map(int, input(f"Enter row {i + 1}: ").split()))
    B.append(row)

# Multiplying matrices A and B
try:
    result = multiply_matrices(A, B)
    # Printing the result
    print("The product of the matrices is:")
    for row in result:
        print(row)
```

```
except ValueError as e:
    print(e)
```

OUTPUT:

```
Enter the number of rows for matrix A: 2

Enter the number of columns for matrix A: 3

Enter the elements of matrix A row by row:

Enter row 1: 1 2 3

Enter row 2: 4 5 6


Enter the number of rows for matrix B: 3

Enter the number of columns for matrix B: 2

Enter the elements of matrix B row by row:

Enter row 1: 7 8

Enter row 2: 9 10

Enter row 3: 11 12
```

Q4 - Write a Python class to check the validity of a string of parentheses. Ensure that the brackets are closed in the correct order (e.g., (), [], {} are valid).

CODE:

```
class ParenthesisValidator:
    def __init__(self):
        # A dictionary to map opening brackets to their corresponding closing brackets
        self.bracket_pairs = { '(': ')', '{': '}', '[': ']' }
    def is_valid(self, s: str) -> bool:
        # Stack to keep track of opening brackets
        stack = []

        # Iterate through each character in the string
        for char in s:
            # If the character is an opening bracket, push it to the stack
            if char in self.bracket_pairs:
                stack.append(char)
            # If the character is a closing bracket
            elif char in self.bracket_pairs.values():
                # If the stack is empty or the top of the stack does not match the closing bracket
                if not stack or self.bracket_pairs[stack.pop()] != char:
                    return False
        # The stack should be empty if all brackets are properly closed
        return not stack

# Taking input from the user
input_string = input("Enter the string of parentheses: ")

# Create an instance of the ParenthesisValidator class
validator = ParenthesisValidator()

# Check if the parentheses are valid
if validator.is_valid(input_string):
    print("The parentheses are valid.")
```

```
else:
    print("The parentheses are invalid.")
```

OUTPUT:

```
 Enter the string of parentheses: {}
The parentheses are valid.
```

Q5 - Write a Python class that reverses a string word by word.

CODE:

```
class StringReverser:
    def __init__(self, input_string):
        # Initialize the string that needs to be reversed
        self.input_string = input_string

    def reverse_words(self):
        # Split the input string into words
        words = self.input_string.split()

        # Reverse the list of words
        reversed_words = words[::-1]

        # Join the reversed words into a single string
        reversed_string = ' '.join(reversed_words)

        return reversed_string
```

```
# Taking input from the user
input_string = input("Enter a string to reverse word by word: ")

# Create an instance of the StringReverser class
reverser = StringReverser(input_string)

# Get the reversed string and print it
reversed_string = reverser.reverse_words()
print("Reversed string:", reversed_string)
```

OUTPUT:

```
Enter a string to reverse word by word: this is aditya
Reversed string: aditya is this
```

Q6 - Write a Python class named Circle with methods to calculate the area and perimeter of a circle, given its radius.

CODE:

```
import math

class Circle:
    def __init__(self, radius):
        # Initialize the circle with a radius
        self.radius = radius
    def area(self):
        # Calculate the area of the circle: A = π * r^2
        return math.pi * self.radius ** 2

    def perimeter(self):
        # Calculate the perimeter (circumference) of the circle: P = 2 * π * r
        return 2 * math.pi * self.radius

# Taking input from the user for the radius
radius = float(input("Enter the radius of the circle: "))

# Create an instance of the Circle class
circle = Circle(radius)

# Calculate and print the area and perimeter of the circle
print(f"The area of the circle is: {circle.area():.2f}")
print(f"The perimeter (circumference) of the circle is: {circle.perimeter():.2f}")
```

OUTPUT:

```
Enter the radius of the circle: 4
The area of the circle is: 50.27
The perimeter (circumference) of the circle is: 25.13
```