```
# Exp11(01)
# Aim:Write a program for cutting tickets and show chair thread.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
from threading import *
from time import *

class Theatre:
    # Constructor
    def __init__(self, str, lock):
        self.str = str
        self.lock = lock

    # Method to repeat for 5 tickets
    def movieshow(self):
        for i in range(1, 6):
            with self.lock:  # Acquire the lock to ensure thread-safe printing
                print(self.str, ":", i)
            sleep(0.5)

# Create a lock for synchronizing the threads
lock = Lock()

# Create two instances of the Theatre class
obj1 = Theatre("Cut Ticket", lock)
obj2 = Theatre("Show Chair", lock)

# Create two threads to run movieshow()
t1 = Thread(target=obj1.movieshow)
t2 = Thread(target=obj2.movieshow)

# Start the threads
t1.start()
t2.start()

# Wait for both threads to finish
t1.join()
t2.join()

# Print thank you message after threads complete execution
print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

# Exp11(Postlb01)
# Aim:Write a program for single thread.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
import time

def task1():
    print("Task 1 started.")
```

```
    time.sleep(2)  # Simulate a task that takes 2 seconds
    print("Task 1 completed.")

def task2():
    print("Task 2 started.")
    time.sleep(3)  # Simulate a task that takes 3 seconds
    print("Task 2 completed.")

def main():
    print("Program started.")
    task1()  # Execute task 1
    task2()  # Execute task 2
    print("Program completed.")
    print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

if __name__ == "__main__":  # Ensure that the program runs only when executed
directly
    main()

# Exp11(Postlab02)
# Aim:Write a program for multiple thread.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
import threading
import time

# Define the first task
def task1():
    print("Task 1 started.")
    time.sleep(2)  # Simulate a task that takes 2 seconds
    print("Task 1 completed.")

# Define the second task
def task2():
    print("Task 2 started.")
    time.sleep(3)  # Simulate a task that takes 3 seconds
    print("Task 2 completed.")

def main():
    print("Program started.")
    # Create two threads for the tasks
    thread1 = threading.Thread(target=task1)
    thread2 = threading.Thread(target=task2)
    # Start both threads
    thread1.start()
    thread2.start()
    # Wait for both threads to complete
    thread1.join()
    thread2.join()
    print("Program completed.")
```

```
    print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

if __name__ == "__main__":
    main()

"""Aim:write an interactive calculator! User input is assumed to be a formula that
consist of a number,
an operator (at least + and-), and another number, separated by white space(e.g. 1
+1).
Split user input using str.split(), and check whether the resulting list is
valid:"""
# Exp05(01)
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
class FormulaError(Exception):
    pass

def calculate():
    while True:
        user_input = input("Enter a formula (e.g., 1 + 1) or 'quit' to exit:
").strip()

        if user_input.lower() == 'quit':
            print("\nName: Sidra Solkar UIN: 231P087 Roll No: 43")
            break

        parts = user_input.split()

        if len(parts) != 3:
            print("Error: Formula must have two numbers and an operator.")
            continue

        try:
            num1 = float(parts[0])
            num2 = float(parts[2])
        except ValueError:
            print("Error: Invalid number input.")
            continue

        if parts[1] not in ['+', '-']:
            print("Error: Invalid operator. Use '+' or '-'.")
            continue

        if parts[1] == '+':
            result = num1 + num2
        elif parts[1] == '-':
            result = num1 - num2

        print(f"Result: {result}")
```

```
# Start the calculator
calculate()

# Exp 10-1
# Write a python Program to send an email to any email address.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43

import smtplib, ssl

port = 587
smtp_server = "smtp.gmail.com"
sender_email = "sidrasolkar2920@eng.rizvi.edu.in"
receiver_email = "sidrasolkar2005@gmail.com"
password = input("Type your password and press enter:")
message = """\
Subject: hello there

This message is send from python."""

context = ssl.create_default_context()
with smtplib.SMTP(smtp_server,port) as server:
    server.ehlo()
    server.starttls(context=context)
    server.ehlo()
    server.login(sender_email,password)
    server.sendmail(sender_email,receiver_email,message)

# Exp 10-postlab1
# Write a program to send multiple emails.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43

import smtplib, ssl

port = 587
smtp_server = "smtp.gmail.com"
sender_email = "sidrasolkar2920@eng.rizvi.edu.in"
receiver_email = ["sidrasolkar2005@gmail.com","sidrashaikh@eng.rizvi.edu.in"]
password = input("Type your password and press enter:")
message = """\
Subject: hello there

This message is send from python."""

context = ssl.create_default_context()
with smtplib.SMTP(smtp_server,port) as server:
    server.ehlo()
    server.starttls(context=context)
    server.ehlo()
```

```
    server.login(sender_email,password)
    for dest in receiver_email:
        server.sendmail(sender_email,receiver_email,message)

"""Write a menu driven python program to perform basic mathematical operations on
two polynomials or integers using numpy."""
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
# Exp 12-1

# A utility function to return maximum of two integers
# A[] and B[] represents coeAicients of first polynomial and second polynomial
respectively
# m and n are sizes of A[] and B[] respectively
def add(A, B, m, n):
    size = max(m, n)
    sum_poly = [0 for _ in range(size)]  # Initialize the sum polynomial with zeros

    # Take every term of the first polynomial
    for i in range(m):
        sum_poly[i] = A[i]
    # Add terms from the second polynomial
    for i in range(n):
        sum_poly[i] += B[i]
    return sum_poly
# A utility function to print a polynomial
def printPoly(poly, n):
    for i in range(n):
        if poly[i] != 0:  # Only print non-zero terms
            if i == 0:
                print(poly[i], end="")
            else:
                print(f" + {poly[i]}x^{i}", end="")
    print()  # Move to the next line after printing the polynomial
# Driver Code
if __name__ == '__main__':
    # The following array represents
    # polynomial 5 + 10x^2 + 6x^3
    A = [5, 0, 10, 6]
    # The following array represents
    # polynomial 1 + 2x + 4x^2
    B = [1, 2, 4]
    m = len(A)
    n = len(B)
    print("First polynomial is:")
    printPoly(A, m)
    print("Second polynomial is:")
    printPoly(B, n)
    sum_poly = add(A, B, m, n)
    size = max(m, n)
```

```
    print("Sum of polynomials is:")
    printPoly(sum_poly, size)
    print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

"""How to get the common items between two python numpy arrays"""
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
# Exp 12-postlab1

import numpy as np
ar1 = np.array([0, 1, 2, 3, 4])
ar2 = [1, 3, 4]
# Common values between two arrays
print(np.intersect1d(ar1, ar2))
print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

"""How to limit the number of items printed in output of numpy array?"""
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
# Exp 12-postlab2

import numpy as np
data = np.array([1, 2, 3, 10, 20, 30])
clipped_data = data.clip(2, 10)
print(clipped_data)
print("\nName: Sidra Solkar \nUIN: 231P087\nRoll No: 43")

# Aim:WAP in python to transpose and find diagonal elements of a matrix
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43
# Exp 13-1

from numpy import *
# accept rows and column
r,c=[int(a) for a in input("Enter rows and column :").split()]

# accept matrix element as a string
str= input(" Enter Matrix Elements :\n")

# convert string into matrix with size r*c
x= reshape(matrix(str),(r,c))
print(" original matrix : ")
print(x)

print(" Transpose matrix : ")
y=x.transpose()
print(y)

print(" Diagonal of a matrix : ")
y=diagonal (x)
```

```
print(y)
print("Sum ofdiagonal : ")
print(sum(y))

# Aim:Write a program to perform transpose of a matrix.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43

def transpose_matrix(matrix):
    rows, cols = len(matrix), len(matrix[0])
    transposed = [[matrix[j][i] for j in range(rows)] for i in range(cols)]
    return transposed

# Function to take matrix input
def input_matrix(rows, cols):
    print(f"Enter elements for a {rows}x{cols} matrix:")
    return [[int(input(f"Element [{i+1}][{j+1}]: ")) for j in range(cols)] for i in
range(rows)]

# Taking input from user
rows = int(input("Enter number of rows for the matrix: "))
cols = int(input("Enter number of columns for the matrix: "))
matrix = input_matrix(rows, cols)

# Compute transpose
transposed_matrix = transpose_matrix(matrix)

# Display result
print("Transposed Matrix:")
for row in transposed_matrix:
    print(row)

# Aim:Write a program to perform matrix multiplication.
# Name: Sidra Solkar
# UIN: 231P087   Roll No: 43

 def matrix_multiplication(A, B):
 # Get the dimensions of matrices
 rows_A, cols_A = len(A), len(A[0])
 rows_B, cols_B = len(B), len(B[0])
 # Check if multiplication is possible
 if cols_A != rows_B:
 raise ValueError("Number of columns in A must be equal to number of rows in
 B")

    # Initialize result matrix with zeros
    result = [[0] * cols_B for _ in range(rows_A)]

    # Perform multiplication
    for i in range(rows_A):
```

```
        for j in range(cols_B):
            result[i][j] = sum(A[i][k] * B[k][j] for k in range(cols_A))

    return result
# Function to take matrix input
def input_matrix(rows, cols):
    print(f"Enter elements for a {rows}x{cols} matrix:")
    return [[int(input(f"Element [{i+1}][{j+1}]: ")) for j in range(cols)] for i in
range(rows)]
# Taking input from user
rows_A = int(input("Enter number of rows for matrix A: "))
cols_A = int(input("Enter number of columns for matrix A: "))
A = input_matrix(rows_A, cols_A)
rows_B = int(input("Enter number of rows for matrix B: "))
cols_B = int(input("Enter number of columns for matrix B: "))
B = input_matrix(rows_B, cols_B)
# Perform matrix multiplication
try:
    result = matrix_multiplication(A, B)
    print("Resultant Matrix:")
    for row in result:
        print(row)
except ValueError as e:
    print(e)
```