Task 1 [30 minutes]

Take a number N as input from the user and create N different threads in the main process. Number each thread from 1 - N. Now divide the threads into a group of four such that:

Group 01 = {1, 5, 9, 13, ...}

Group 02 = {2, 6, 10, 14, ...}

Group 03 = {3, 7, 11, 15, ...}

Group 04 = {4, 8, 12, 16, ...}

The thread in each group will execute within 1 second (All threads in the group are executed within 1 second) and then move on to the second group.

The output of each thread is

I am thread No. x with thread id y.

Sample Output:

```
Enter N (No. of threads) : 9
Creating 9 threads

After 1 second
I am thread No. 01 with thread id xxxxxxxxxx
I am thread No. 05 with thread id xxxxxxxxxx
I am thread No. 09 with thread id xxxxxxxxxx

After 2 seconds
I am thread No. 02 with thread id xxxxxxxxxx
I am thread No. 06 with thread id xxxxxxxxxx

After 3 seconds
I am thread No. 03 with thread id xxxxxxxxxx
I am thread No. 07 with thread id xxxxxxxxxx

After 4 seconds
I am thread No. 04 with thread id xxxxxxxxxx
I am thread No. 08 with thread id xxxxxxxxxx

Main was waiting for the threads and is now terminating....
```

Task 2 [30 minutes]

Write a program that performs matrix multiplication using threads to improve performance on multi-core processors. The program should take two input matrices, A and B, and calculate their product, C. The matrices A and B should have dimensions suitable for multiplication. The program should utilize a specified number of threads (2) to parallelize the multiplication process. Each thread should be responsible for computing a portion of the result matrix, and the result should be stored in matrix C. Ensure that the program correctly divides the task among threads. After multiplication, the program should print the resulting matrix C. Adjust the dimensions of the matrices and the number of threads as needed.

Task 3 [50 minutes]

You will do Forward propagation of Neural Networks using Multiprocessing and multithreading.

Explanation of Forward Propagation of Neural Networks:

It consists of 3 steps,

Step 1: multiply the weight matrix W with the features matrix x_i.

Step 2: add the result of step 1 with biases matric b. the result will be called Z.

Step 3: Apply the sigmoid function on all elements of the resultant matric from step 2.

Sigmoid function is $A=1/(1+e^{-z})$

So, create three processes P1, P2, and P3. P1 will create multiple threads to multiply the weight matrix and biases matrix. (Matrix Multiplication using multithreading). The number of Threads in P1 will be equal to the number of rows of weights W.

P2 waits for its completion and uses the resultant matrix to add its biases matrix using multithreading (Matrix Addition using Multithreading). The number of Threads in P2 will be equal to the number of rows of biases matrix b.

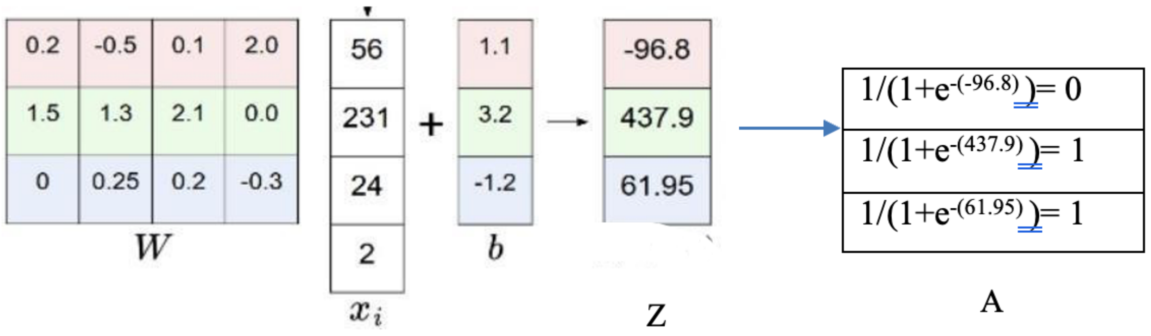
P3 weights for its completion and apply the sigmoid function on all elements of the resultant matrix from P2. P3 will also use multithreading to apply the sigmoid function. The numberof Threads in P3 will be equal to the number of rows, resulting from P2. All threads work simultaneously to calculate the sigmoid function.

Hints:

- Use exp function present in cmath library to compute exponent. For example, exp(5) will calculate e^5.

- Use pow function present in cmath library to compute the power. For example pow(4,5) will calculate 4^5.

Sample Example:


0.2
-0.5
0.1
2.0
1.5
1.3
2.1
0.0
0
0.25
0.2
-0.3
W
56
231
24
2
x_i
+
1.1
3.2
-1.2
b
-96.8
437.9
61.95
Z
1/(1+e^-(-96.8))= 0
1/(1+e^-(437.9))= 1
1/(1+e^-(61.95))= 1
A