NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES ISLAMABAD

OPERATING SYSTEMS Spring 2022

ASSIGNMENT 03

Due Date: 11:59 PM, 2 May 2022.

Instructions

- Zero marks will be awarded to the students involved in plagiarism.
- All the submissions will be done on Google Classroom.
- You have to submit .c/.cpp files. The naming convention has to be followed Each question will be named as q1.cpp/q1.c. You have to submit 1 zipped file having the questions.
- Be prepared for viva or anything else after the submission of the assignment.

QUESTION NO. 01:

You are required to do multithreading on generated receipt. You will create 5 threads.
Thread 1: select the number of items you purchased. Display the prices of all items as (Quantity x Price of the individual) add them, and return their sum. Thread 3: Calculate 8% tax on it.
Thread 4: If an order is greater than 250, Calculate 10% sale.
Thread 5: Sort the items according to their prices.

Output: Thread 1: Items Purchased = eggs, bread, chocolate
eggs = 3 x 15=45 , bread = 1x 60 =60 , chocolate = 5 x 50 = 250, sum=45+60+250= 355 Thread 2 : 355/(1+0.8) = 197.22 , taxed sum=197.22+355= 552.33 Thread 3 : price=355 , sale =0.1 , 355*0.1=35.5 , 355-35.5 = 319.5
Thread 4:

Items	Price
Chocolates	250
Bread	60
eggs	45

QUESTION NO. 02:

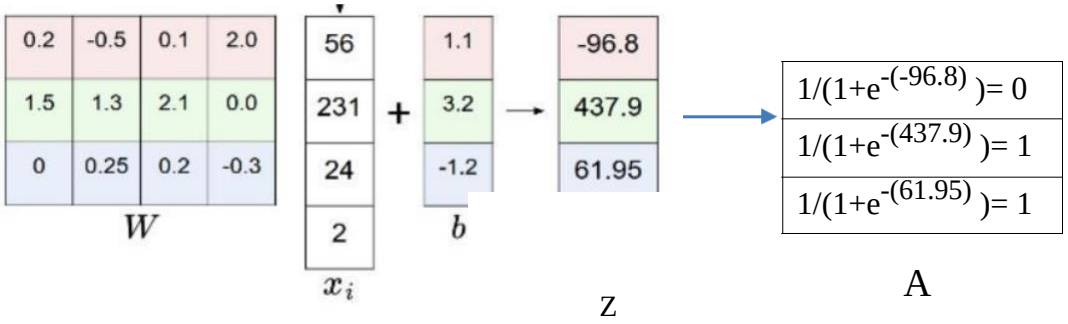
You will do Forward propagation of Neural Networks using Multiprocessing and multithreading.

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



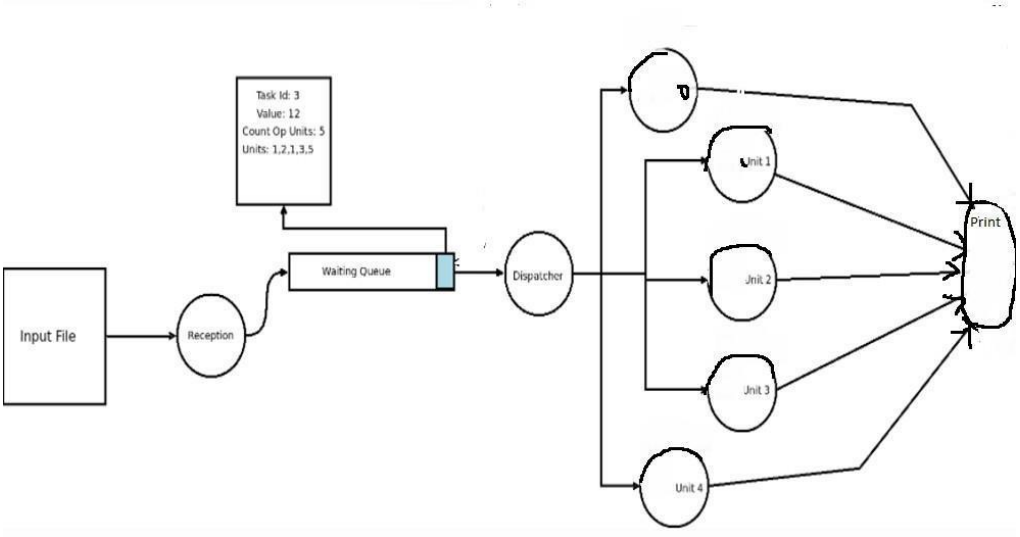
So, create a process P1, P2, and P3, p1 will create multiple threads to multiply the weight matrix and biases matrix. (Matrix Multiplication using multithreading). The number of Threads in p1 will be equal to the number of rows of weights W.

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 number of Threads in p3 will be equal to the number of rows, resulting from p2. All threads work simultaneously to calculate the sigmoid function.

QUESTION NO. 03:

Consider a complex system, such as a workplace, with three rooms (Reception, Dispatcher and Print) and processing units that perform multiple duties.



This workplace has a reception room which is responsible for putting incoming tasks into a waiting queue. The dispatcher room is responsible for sending these tasks to the corresponding unit of execution.

Task structure includes its ID, value, arrival time, number of units, units assigned and total number of unit.

3.1 Input File

The structure of input file is as follows:

TaskId	TaskVal e	UnitsCount	UnitId1	UnitId2	UnitId3	UnitId4	UnitId5
0	123	5	4	1	3	0	2
2	-3	4	3	2	1	0	
1	78	3	1	2	4		
3	9	2	0	4			
4	100	1	2				

3.2 Reception Room

The Reception room should read an input file and make a structure of each task, and determine the arrival time of the task, then put that task into the waiting queue.

```
struct task {
   int id;                  // id of this task
   int task_value;          // value of this task
   int arrival_time;        // the arrival time of the task
   int unit_count;          // number of units
   int *unitIdList;         // list of units which task will be assigned to
}
```

3.3 Dispatcher Room

A dispatcher should operate on the waiting queue and do the following:

1. Get the tasks from waiting queue in FCFS order
2. Invoke the required number of units(Threads) mentioned in the task structure as unit_count
3. Get the Unit Ids from unitIdList
4. Pass the task structure to the corresponding unit of execution obtained from unitIdList

3.4 Unit Operations

Each unit of execution performs its own operation for the given **task value**
For instance, unit 0 adds 5 to the task's value and then modulates the new number with S.

- S is a constant value of 5000 that should be defined in your program.

Each of these units are implemented as threads. Furthermore, these units have their own queues and these queues are filled by the dispatcher

Here is the detail what each unit is doing

Unit id	Operation	Description.
0	+5, %M	Adds 5 to the value then modulate by S
1	*7 %M	Multiplies value by 7 then modulate by S
2	^5, %M	Calculates the power 5 and then modulate by M
3	-100	Subtracts the 100 from value
4	^2	Takes the square of value

3.5 Print Room

The print room will print the task id, completion time of task, number of units involved in completing the task and task value generated by each unit .This Room work as below:

- Receives TaskValue computed by corresponding Unit
- Combines the UnitId with TaskValue through "-"
- Prints the TaskValues in the FCFS order computed by the Units.

The format of output should look like the following.

TaskId	CompletionTime	UnitIds	TaskValue1	TaskValue2	TaskValue3	TaskValue4	TaskValue5
0	in milliseconds	4,1,3,0,2	unitId-reuslt	unitId-reuslt	unitId-reuslt	unitId-reuslt	unitId-reuslt
1	in milliseconds	3,2,1,0	unitId-reuslt	unitId-reuslt	unitId-reuslt	unitId-reuslt	
2	in milliseconds	1,2,4	unitId-reuslt	unitId-reuslt	unitId-reuslt		
3	in milliseconds	0,4	unitId-reuslt	unitId-reuslt			
4	in milliseconds	2	unitId-reuslt				

- To get the arrival time and the duration of being in the queue you can use clock_gettime or gettimeofday.
- You are not allowed to use any IPC mechanism (Named,Unnamed Pipes, shared memory) to pass the Task structure from Dispatcher to Units.

Good Luck!