Assignment 2

***Instructions****:*

1. *Copied assignments will be marked negatively.*
2. *Late submissions will not be entertained in any case.*
3. *LLMs can be used but your understanding will be evaluated through demos following the due date.*

# Q1:

You are required to do multithreading on generated receipt. You will create 4 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 2: Calculate 8% tax on it.

Thread 3: If an order is greater than 250, Calculate 10% sale.

Thread 4: Sort the items according to their prices.

**Output:**

Thread 1:

Items Purchased = eggs, bread, and 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 |

**Deliverable:**

C/C++ code for the above scenario with outputs.

# Q2:

# Travelling Salesman Problem (TSP): Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

# For example, consider the graph shown below. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

# A diagram of a triangle with numbers and circles Description automatically generated

# The data is provided in file showing distance between all the nodes. The number of nodes can vary but make sure to attach the testFile while submitting your code.

# testFile.txt

# 4

# 1 - 2 10

# 1 - 3 15

# 1 - 4 20

# 2 - 3 35

# 2 - 4 25

# 3 - 4 30

# Take a number N from user and create N different threads. Each thread will find a solution of TSP by randomly generating numbers (Make sure the sequence of random numbers generated by different thread is not the same). Now each thread will save their output (cost of the tour) in a global array (after displaying it on terminal along-with their thread id). When all threads will be done computing the value, they will check whether their value was the minimum cost or not. The threads whose solution was not shortest will be stuck in an infinite loop. The one with shortest solution will output the thread id and cancel all the remaining threads.

# If two threads have the same value, then define a mechanism as tiebreaker and explain it in comment on the top of your code file.

**Deliverable:**

C/C++ code for the above scenario.