ONUR DİLSİZ
2019400036

Project#3
Simulation of a Fun Fair Payment System
BOFUN Fair - It's Fun!

There are several global variables for this project, including the balances of different companies, the customer number and its counter, and mutexes to synchronize threads. There are three uses of mutexes in this implementation. The first is mutexesForCustomers[10], with one mutex per machine. This mutex is locked when a customer provides data to a shared vector, and then it is unlocked after the related vending machine uses it. The second mutex is for the companies, to prevent concurrent additions and in order to produce the correct output. The last mutex is used to prevent simultaneous writing to the output file.

The program requires an input file, which is provided as a command line argument. The input file is read and the lines are stored in a vector called 'lines'. From these lines, 'customerStruct' objects are created, which contain a customer's ID, sleep time, ticket vending machine preference, prepayment company name, and amount.

Then, vending machine threads and customer threads are created and joined in the main thread. The main thread writes the necessary lines to the output file.

Customer Thread Function:

In the customer thread function, the parameter is a void pointer which is converted to a struct pointer at the start of the function. After the conversion, the sleep time is taken from the customer struct and the thread sleeps for that amount. The customer then locks the mutexesForCustomer to prevent another customer from using the same vending machine. The data in the shared memory is then changed.

Vending Machine Thread Function:

In the vending machine thread function, the parameter is a void pointer which is converted to an integer pointer at the beginning of the function. The integer represents the vending machine's ID. There is a loop that provides busy waiting for the vending machine until the last customer finishes their payment. When there is data in the shared vector, the vending machine thread gets the data from the shared vector. Calculations and log operations are performed using this data, and then the data is erased from the shared vector and the related mutexesForCustomer is unlocked. Before the log operation, the mutexForLog mutex is locked, and after the log operations it is unlocked. Similarly, calculations have their own mutex per company, which is locked before changing the value and unlocked afterward. The customer's payment amount is then added to the relevant company.