Bahri Alabey 2019400228

This code is a program that simulates the functionality of vending machines and the sale of tickets through these machines. The program is implemented using pthreads, which allow for the creation and management of threads in C.

The program begins by defining a struct called `customer_info` that stores information about customers and their purchases. This struct contains four fields:

- `customer_id`: an integer representing the unique ID of a customer
- `sleep_time`: an integer representing the amount of time a customer spends at a vending machine
- `vending_machine_id`: an integer representing the ID of the vending machine the customer will be using
- `vending_company_name`: a char pointer representing the name of the company that owns the vending machine
- `payment_amount`: an integer representing the amount of money that will be paid by the customer

The program also defines a struct called `vending_info` that stores information about the vending machines and their sales. This struct contains three fields:

- `fp`: a file pointer that points to a file where the program writes the output
- `vending_id`: an integer representing the ID of a vending machine
- `total_customer_number`: an integer representing the total number of customers that will be served

The program also defines several global variables. `VENDING_MACHINE_NUMBER` is a constant integer representing the total number of vending machines in the program. `TOTAL_COMPANY_NUMBER` is a constant integer representing the total number of companies. `mutexQueue` is an array of `pthread_mutex_t` objects, one for each vending machine, that are used to synchronize access to the task queue of each vending machine. `condQueue` is an array of `pthread_cond_t` objects, one for each vending machine, that are used to signal waiting threads when a queue of a vending machine becomes available. `taskQueue` is a two-dimensional array of `customer_info` structs, with `VENDING_MACHINE_NUMBER` rows and 300 columns, that stores the tasks for each vending machine. `task_count` is an array of integers, one for each vending machine, that stores the number of tasks in the task queue for each vending machine in a FIFO queue manner. `total_balances_of_companies` is an array of integers, one for each company, that stores the total amount of money earned by each company. `total_sales_on_companies` is an array of integers, one for each company, that stores the total number of sales made by each company. `mutexBalances` is an array of `pthread_mutex_t` objects, one for each company, that are used to lock balance of a company when it's updated.

The program contains several functions: mainly `customer_func` and `sell_ticket` `vending_machine_func`.

`vending_machine_func` is a function that is executed by a vending machine thread when it is created. The function takes a single argument of type `void *`, which is a pointer to a `vending_info` struct that contains the information about the vending machine. The function enters waits for customers to enter its queue. `pthread_cond_wait`. When a customer arrives at the machine, the function removes the customer from the queue and calls `sell_ticket` to process it. It takes the information from the queue using the customer struct I've wrote. The function then continues to wait for the next task.

`customer_func` is a function that is executed by a customer thread when it is created. The function takes a single argument of type `void *`, which is a pointer to a `customer_info` struct that contains the information about the customer. The function first casts the void pointer argument back into a `customer_info` struct and stores it in a local variable called `customerInformation`. It then sleeps for `sleep_time` milliseconds using `usleep`.

The `customer_func` function continues by acquiring a lock on the mutex associated with the vending machine specified in the `customerInformation` struct using `pthread_mutex_lock`. This ensures that only one customer can access the task queue of the vending machine at a time. The function then adds the `customerInformation` struct to the task queue of the vending machine and increments the task count for the vending machine. The function then releases the lock on the mutex using `pthread_mutex_unlock` and sends a signal to any waiting threads using `pthread_cond_signal`. This allows the vending machine thread to wake up and process the task in the task queue. Finally, the function exits using `pthread_exit`.

The `sell_ticket` function is a helper function that is called by the vending machine threads to process tasks in the task queue. The function takes three arguments: a `customer_info` struct containing information about the customer, an integer representing the ID of the vending machine thread, and a file pointer pointing to a file where the sales information is written. The function first writes the sales information to the specified file using `fprintf`. It then updates the balance and sales information for the company that owns the vending machine using one of the global variables defined earlier. If the company name is "Kevin", "Bob", "Alice", "Tom", or "Jerry", the function acquires a lock on the mutex associated with the company using `pthread_mutex_lock`, updates the balance and sales information for the company, and releases the lock using `pthread_mutex_unlock`. Finally, the function sends a signal to any waiting threads using `pthread_cond_signal` to the threads that are waiting for that lock to be unlocked.