CSE344 HW4 Report
161044070
Kaan Can Bozdoğan

Synchronization Requirements and Solutions to Them:

Make the homework queue available for only one thread at a time. Because both main and G thread makes alterations in it. Semaphore **sem_q**.

There is a producer - consumer relationship between main and the G thread because while G thread adds homework to the Queue, the main thread removes them from it. So a semaphore to count the available homework in the Queue is needed. Main thread waits for it. G thread posts it. Semaphore **hwCount**.

In order for main to assign a homework to a student, there needs to be at least one student not busy doing a homework. A semaphore to keep the count of available students is needed. Main thread waits for it. Student threads posts it when they finish a homework. So there is always a student available waiting for homework assignment. Semaphore **avlStdCount**.

Main thread reduces the budget after it chooses a student to assign a homework. It is not reduced in the student thread because main needs to know the reduction in the budget immediately, so it can react to budget change properly. And student doing a homework is printed in the student thread. Sometimes the G tread works after homework is assigned to a student but before the student prints the information of itself doing the homework. So money is reduced but the user don't know the student doing the homework. So G thread prints the reduced budget before the user of the program recognising a student did a homework. So in order to prevent that a semaphore is needed. Just before reducing the budget, main thread waits for it. And after student thread posts it after it printing the knowledge. It does not make the program work slower. Just makes student thread work right after a homework is given to it. Semaphore **print**.

A student thread needs to be notified when a homework is given to it. And the type of homework just to print it. A pipe is needed. Because pipe makes the thread to wait without making the cpu busy and can transfer the homework type as a character between threads. And needs to be unique for every student so an array is needed. Pipe **p_stds[]** with the size of student count.

Other Requirements:

A **Queue** struct. A **QueueNode** struct. Queue add function to add a homework to the end of it. Queue remove function to remove a homework from the head of the queue and return the removed homework. Is empty function. Queue clean function to clean the allocated spaces in the queue at the end of the program.

A **Student** struct to hold the data given in the student file and the information about if the student is available for a new homework, count of homework finished and the index in the student array to find its corresponding pipe from the pipe array etc.

Main chooses students who are best fit according to the homework type. I thought checking every student every time to understand which student is the best fit is not optimal. So I made arrays for every homework type which have students already sorted in them starting from the best fit for to worst. Main thread starts checking students from the beginning of the array to see if they are not busy doing work and gives the homework to the first available student in the array. So instead of sorting every time I sorted 3 arrays at the beginning. So a sorting function is needed.

How threads work:

G thread:
Gets the pointer to the Queue as an argument.
Reads a homework from the file. Waits for sem_print, prints the homewrok it read, posts sem_print. Waits for sem_queue, adds the homework to the queue, posts sem_queue and sem_stdCount. Loops until end of file or sig SIGINT arrives or budget is not enough.

Student thread:
Gets the pointer to the corresponding student struct as an argument.
Prints it is waiting for a homework. Waits for a homework from the pipe. If coming homework is not -1 prints that it is doing the homework, posts sem_print and sleeps. After sleeping, changes its busy status to 0 and posts sem_avlStdCount. Loops until a SIGINT arrives or given homework through the pipe is -1.

Main:
Read student information to student array. Creates sorted arrays for each homework type. Initializes semaphores and pipe array. Creates the Queue. Creates G thread and student threads.
Starts looping until budget is not enough for any available student OR Queue is empty while the end of homework file is reached OR SIGINT arrives. In the loop:
Waits for sem_avlStdCount, waits for sem_queue, removes a homewrok from the array and posts sem_queue. Waits for sem_avlStdCount. If SIGINT has not arrived yet, chooses the best student from the available students from the corresponding sorted array and sends the homework as character to the corresponding student thread's pipe. If it can not find any students to give the homework to, it means that there is not enough money for any available students.

After the homework solving loop ended:
If SIGINT arrived cancels all the other threads. If not waits for every student to be available and sends -1 through their pipe so student treads can quit out of the loop and finish executing. Prints the information about how many homework the students solved and how much money they made and remaining money in the G’s account. Close the files, join the threads, destroy the semaphores, clean the Queue and free every allocated space and finish the program.