Musical Chairs Report

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand our responsibility to report honour violations by other students if we become aware of it.

Name: Bandi Chiranjeevi Sai Ganesh, Navudu Jyothi Swaroop

Date: 24-02-2020

Signature: B.C. Sai Ganesh,N.J.S

I have used c++ threads and synchronisation primitives like mutex locks, condition variables.

<u>In musical_chairs function:</u>

I have created array A which stores player id of the players who lost the game.

I have created an array slep which is used to store the sleep quantum of player threads.

I have created a thread to run umpire_main function. After that I used join function to make the thread to wait for the umpire thread until it finishes it’s job.

After that it prints the output in the required manner as given in question by using the values in the array A.

In umpire_main function:

I have initialised the variable no_of_chairs (which is declared globally) to no. of players-1. And then I have created n player threads. Umpire thread reads the input from file.And when it reads " player_sleep" the umpire thread will read another two values, they are player id and sleep quantum, sleep quantum of player is stored in slep array at index player id.And when it reads "umpire_sleep", it sleeps for specified amount of time. And when it reads "music_stop" string, umpire thread gives control to player threads by calling notify_all() function. umpire will wait on a conditional variables, until all the current players finishes their job. In every lap the last player (who lost the game) will call notify_all() function on the conditional variable in which the umpire thread is waiting. After that the umpire updates the no. of chairs in the variable no_of_chairs and makes all the entries in slep array to zero.

Umpire thread waits until all the players finishes their job(either win or lose) by using thread join function.

In player_main function:

I have used unique mutex lock to maintain data consistency in other words to allow only one player to critical section.

I initialised infinite while loop which has break statament.In that loop.

Every threads look at the value in slep array using player id as index,if the value is zero it proceeds otherwise it will sleep for sometime which is speficied at the index of slep array.

Every thread first checks if the no. of chairs available is zero or not. If the no. of chairs available is not zero, then player thread occupies the chair(reduces the value in no_of_chairs variable by one) and will wait on conditional variable thereby

releasing the lock.This process is continued until all the chairs are filled(until value of no_of_chairs becomes zero) and the last player who last the game will enter his id in the array A and call notify_all() which makes umpire thread to wake up and this player thread job is over(uses break statement to come out of loop).

I have used sleep_for function to make threads to sleep for a specified time.

Winner player id is collected using the winner(int type) variable.

After execution of the umpire thread,the thread which created the umpire thread will print the result using array A and the winner variable.