STDISCM

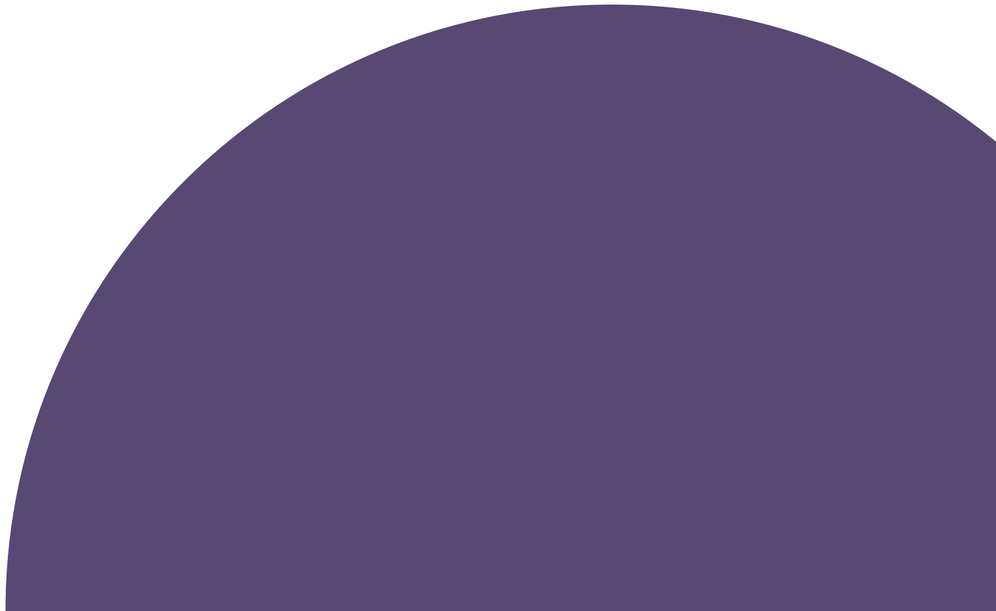
S11 - PSET2

Chan, Dane Marcus

https://github.com/TheBanana03/dungeon_allocator.git

DEADLOCK

- A deadlock can happen depending on the program design. If a dungeon thread is holding a resource while waiting for a party to be assigned to it, but the main thread is waiting for that same resource before releasing the party to be assigned to that thread, it could lead to a deadlock.



DEADLOCK

A deadlock cannot happen because:

- When threads get woken up, it makes the edits in a mutex, which means that it would hold the resource
- However, it does not wait when editing the shared resource, and simply edits it immediately
- This results in no deadlock, because a thread will always make use of its shared resource immediately without waiting for other resources, and will explicitly not be holding resources while waiting for another party to arrive.

STARVATION

- Threads may stay idle while others hoard processing time, leading to the starvation of other threads. This happens if particular dungeons are repeatedly selected; for example, a party dispatcher might always assign a party to the first empty thread in the array, neglecting the threads farther down the array, which can happen when a party clears a dungeon very quickly.

STARVATION

- The code solves this by having each thread have the responsibility of looking for a new part through the use of semaphores. Since each thread uses a semaphore to wait for a party, they get placed in a queue, which would mean that, eventually, all of the threads will be rotated through, diminishing the chances of starvation.