Andrew Tran

Project 2 Report

In my solution I made sure that any resource that was needed by the other process was released before blocking on another process. For example in the initial stage when a guide and visitor are needed to open the museum, the visitor calls up() on the resource that the guide requires (waitV) before calling down() on the resource it needs itself (waitOpen). By making sure both processes are not holding any resources before they block on another resource they need, dead locks are prevented. Starvation should not occur in this solution because there is a limit to how many visitors can be in the museum at a time. Once the limit it reached, those visitors must take their tour and return their resources before more visitors can enter. This guarantees that all visitors will have a chance to run no matter how many visitor processes are created because at least 1 visitor should run for every 10 visitors that arrive. All processes that are blocked for whatever reason are put into a priority queue in a semaphore using task_nice() to determine their priority. Since no processes are treated any differently in terms of choosing their priority, this leads to a fair solution.