PROJECT-2 / Santa's Workshop

Important: All comments are made in the c file in PartC-D folder.

Instruction: In order to run the code, execute "gcc project_2.c" or "gcc project_2.c -lpthread" in your terminal (depending on your OS). Then, run "./a.out -t DESIRED_SECONDS -s DESIRED_SEED -n DESIRED_TIMING", where parameters indicates the running time of the program, desired seed for random numbers and desired timing that gives all of the queues once in a second after that specific second.

1) Part-I

This part's code is separated in a folder named "Part-A". Part-A is a base for other parts, where we create our structure and start to run the program. In this part, we create queues, mutexes, arrays (to check the P/Q and A/Q jobs) and other int variables to manipulate the code as desired.

As the main starts to run, initializations are completed in init() function. After that, with a timer checktime(), every second a new gift arrives, depending on probability stated in the pdf. Our create_request() function accomplishes that, assigning probabilities to numbers between 0 and 20 (excluded). Created requests are added to the corresponding queues.

Elf-A, Elf-B and Santa accomplishes the corresponding tasks within a while loop. This while loop has "If-Else" conditions to provide desired priority between jobs. Packaging jobs and delivery job are prioritized among other jobs to reduce waiting time. "If-Else" conditions are structured by size of the queues, in other words if a queue has at least one item, it means that there is a job to do. After timer indicates that the limit is reached (given by command line argument, -t NUMBER), while loop in the main is forced to break. Finally, threads and mutexes are killed and program stops.

2) Part-II

Structure may remain same since only change in Santa's priority. Therefore, we added an "If-Else" condition in Santa's function to provide desired priority, which forces Santa do QA if one of the two conditions (stated in PDF) are satisfied.

This is done via adding the code below at the very beginning of the while loop of Santa;

if ((delivery_q->size > 0) & (qa_q->size < 3))

3) Part-III/IV

First, we added an "IF-ELSE" condition in our main while loop that implies that modulo formula, so that every 30^{th} second a gift to New Zealand with highest priority arrives. In that second, also a regular gift arrives too. In order to implement gifts to New Zealand, we created analogous queues of the regular queues. Same process of creating gifts is applied

to the gifts to New Zealand in a new function called **“create_zealand()”**. In worker functions, the priority of gifts to New Zealand is also applied with “If-Else” conditions. Every worker function has an “If” at the very beginning of their while loops that inspects the New Zealand queues. If there exists a queue of them that needs to be done, then the worker immediately starts to execute that job, before starting the regular tasks. The tasks with multiple stages inform each other by changing an boolean variable in an array, designed specific to that duty. Task is sent to the responsible next queue by **“send_next()”**. This function also updates the **task_arrival** attribute of the task, so that **turnaround** can be calculated. In order to keep the logs, we designed a writer function, which has three parameters; task, task_type and worker. When a job is done, its corresponding attributes are calculated and inserted into the log file. Finally, all of the queues are printed out after n^{th} second, where n is a command line argument.