COMP 304: Project 2
Santa's Workshop

Due: Dec 13th, 11.59 pm

Notes: The project can be done **individually or in teams of 2**. You may discuss the problems with other teams and post questions to the OS discussion forum but the submitted work must be your own work. This assignment is worth 14% of your total grade.

Contact TA: Ilyas Turimbetov (ENG 230) iturimbetov18@ku.edu.tr

Description

In this project, you will get more familiar with the concepts of scheduling, synchronisation, multi-threading and deadlock prevention in operating systems by using POSIX threads (pthreads) API.

Santa's workshop

You have been hired by Santa to verify whether he can complete his job this Christmas with only 2 employee elves left due to economical crisis. Your job is to implement a safe and deadlock-free simulator of his workshop. His workshop operates as shown on the image below. There are 3 types of gifts: just chocolate, toy and a chocolate and for the best kids a one with also a GameStation on top. Every present needs to be packaged by one of the elves and delivered by Santa. Santa's workshop is unable to produce anymore, so all the gifts are manufactured in China. Toys require additional customization, and due to being expensive, GameStations need to pass quality assurance (QA) by Santa as shown on the image.

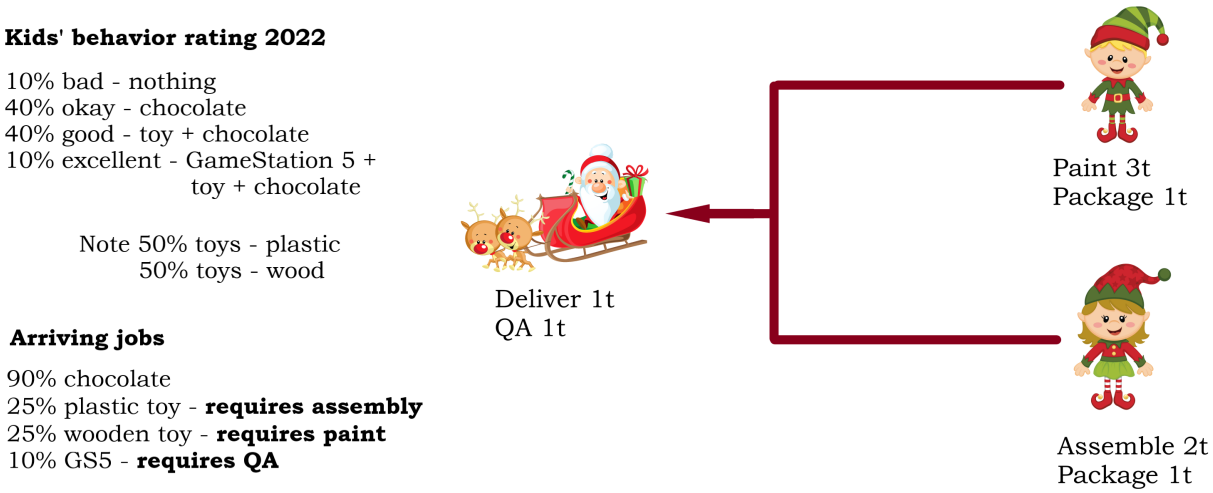

Figure 1: Santa's workshop

Part I

(50 points) Here is how the workshops simulation is going to work. Every t secs (it is assumed to be 1 sec) there is a 90% probability that a gift request arrives. From the tables on Figure 1 you can see that every t seconds

1. There is a 40% chance that a gift (containing only a chocolate) will only need packaging, followed by delivery.
2. There is a 20% chance that a gift (containing a wooden toy and a chocolate) will require painting, followed by packaging, followed by delivery.
3. There is a 20% chance that a gift (containing a plastic toy and a chocolate) will require assembly, followed by packaging, followed by delivery.
4. There is a 5% chance that a gift (containing a GS5, wooden toy and a chocolate) will require painting and QA, followed by packaging, followed by delivery.
5. There is a 5% chance that a gift (containing a GS5, plastic toy and a chocolate) will require assembly and QA, followed by packaging, followed by delivery.

The same information can be represented in a form of Directed Acyclic Graphs (DAG). You can see from them, that in gifts of type 1, 2 and 3 all tasks are strictly one after the other. In those of type 4 and 5, however, QA can be done by Santa at the same time as Painting or Assembly is done by one of the elves. Note that Packaging only has to start after both of the previous tasks are completed.

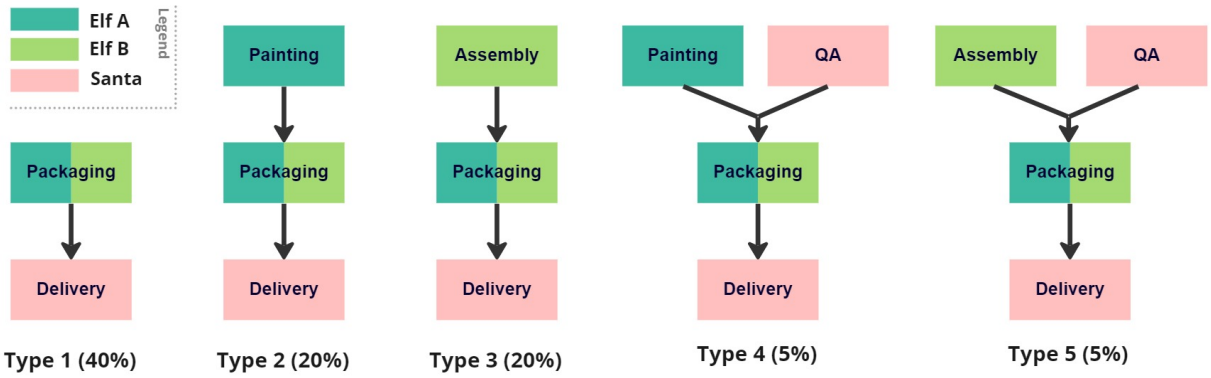

Figure 2: Task DAGs for every gift type

Here are more detailed rules for the simulation environment.

- The simulation should use real-time. Get the current time of the day, run the simulation until current time + simulation time.
- Elf A can paint wooden toys or package gifts. Elf B can assemble plastic toys and package gifts as well. Santa handles delivery of packaged gifts and quality check of GameStations.

- There is only one job being done by each worker at a time.
- To reduce the waiting time of gifts, elves always prioritize packaging tasks, while Santa prioritizes delivery.
- Each painting task takes $3t$ secs to complete. Assembly takes $2t$ secs, packaging, delivery and QA take $1t$ secs.
- Use a command line argument *-s* to indicate the total simulation time (e.g. -s 200).

If we were being realistic, t would be in the order of minutes but assume t is 1 sec for the purpose of the computer simulation.

Part II

(30 points) Part I may cause starvation to the QA tasks that Santa needs to perform. Since he needs to prioritize delivery, QA may be indefinitely delayed. If you have implemented Part I correctly though, you will notice that this is not the case. However, Santa has been contacted by UNICEF, which considered his rating system unethical and forced him to give presents to all kids. Therefore, the 10% of kids with bad behavior need to get a chocolate too. Try to run your simulation with such a setting (Gifts of type 1 will have 50% chance instead of 40%). Now, you will see that starvation becomes a problem. To be able to give best gifts to the kids with excellent behavior and not leave everyone out, you will need to implement a solution to the starvation of QA tasks. You will need to make Santa do QA if one of the two conditions is satisfied:

- (a) No more presents are waiting to be delivered,
- (b) 3 or more GameStations are waiting to go through QA.

Part III

(10 points)
Now, Santa realizes that kids from New Zealand need to get their presents urgently, because they are too far and they can't get the gift on time otherwise. Assume that every $30t$ there is a gift request arriving from New Zealand and give it priority for every task it requires. Note: The task that is already being processed can be completed.

Keeping Logs

(10 points)
You are required to keep a log for the events and for each worker (elves and Santa), which will help you debug and test your code, as well as help to understand if Santa's plan can be implemented. The events.log should keep the task no (ID), gift no (ID), gift type (as in Figure 2), task type (painting (P), assembly (A), QA (Q), packaging (C), or delivery (D)), the time when the gift request arrived, the time when the task arrived, turnaround time

(TT) of the task (end time - arrival time), and who performed the task (Elf A, Elf B, or Santa (A, B and S)). From the logs you can easily confirm whether the gift has passed all the needed stages before being delivered. (For example, task of type 5 from the list in Part I should pass assembly(A), QA (Q) (these can happen at the same time), then packaging (C), then delivery (D).

TaskID	GiftID	GiftType	TaskType	RequestTime	TaskArrival	TT	Responsible
1	1	3	A	0	0	2	B
2	2	1	C	1	1	1	A
3	3	4	P	2	2	3	A
4	2	1	D	2	1	1	S
5	1	3	C	2	0	1	B
6	4	2	P	3	3	5	A
7	1	3	D	3	0	1	S
8	3	4	C	3	2	1	B

Output the snapshot of the waiting tasks with their IDs in every second starting from n^{th} secs to the terminal, where n is also a command line argument. The numbers indicates the task IDs waiting in the queue at time n. Feel free to come up with a better representation or GUI.

At 30 sec painting : 33, 35, 36
At 30 sec assembly : 32, 37
At 30 sec packaging : 23, 24, 28, 31, 33
At 30 sec delivery : 25, 26, 27, 30
At 30 sec QA : 12, 21

Note: the sample outputs given to you are just for demonstration and do not represent the expected output

Implementation

- The starter code given to you outlines a possible structure of your project, but you are free to make changes.
- You may want to keep a queue for each task type, add tasks to queues at the appropriate times. The queue code is provided to you, but you can edit it if you want to, especially the Task objects.
- You can use random number generator to generate gifts at the specified probability. For easy debugging, use the same seed.
- You should represent each worker as a thread. Each worker's thread will sleep for a given amount of time to simulate a task being performed.
- Start simple. For example simulate a Santa and a single Elf with only gifts of type 1. Add complexity as you are sure the current implementation works (no deadlock, no race condition).
- To sleep pthreads, please use the code provided to you. Do not use sleep() system call.
- For Pthread semaphores, mutexes and condition variables, also refer to the pthread tutorial online: https://hpc-tutorials.llnl.gov/posix/
- Revisit the condition variable example we covered in the class.

Deliverables

You are required to submit the followings packed in a zip file (named your-username(s).zip) to blackboard :

- .c or .cpp source file that implements the simulation. Please comment your implementation.
- sample log files for 120 sec simulation, with random seed 10.
- any supplementary files for your implementation if needed (e.g. Makefile)
- a README (report) file describing your implementation, particularly which parts of your code work. Since we cannot hold a demo with everyone, document your report properly, but keep it clear and short.
- Finally you may perform a demo if requested by the TA if there are issues with your implementation.

GOOD LUCK.