Task 3

Report

Liubov Mikhailovna 60204 and Jorge Fresco 60209

Concurrency

- **Concurrency is ensured by modeling each conveyor belt and truck as independent Erlang processes** using spawn, with all processes operating concurrently. The package generator (*generator_loop*), conveyor belts (*conveyor_loop*), and trucks (*truck_loop*) work in parallel without interference.
- Each conveyor belt has an independent queue of packages received from the generator, communicating with trucks via message passing, while trucks maintain their own state (remaining space), accepting packages until full or out of space for the next package and replacing themselves, taking between 1 to 3 seconds to reset their remaining space.

Deadlock-Free Operation

- The system avoids deadlocks by ensuring that all processes don't block infinitely, with asynchronous message handling and making sure no packages are lost.
- **Conveyor belts** either periodically process their queue by sending a package to its truck to be loaded or buffer new received packages for future processing in their queue. If a package doesn't fit into the truck, it is retrieved and put back into the queue again to be processed later after the truck is replaced. When a truck is being replaced, it receives a message from the truck *{pause_operation}* to pause all operations and only resume when receiving another message from the truck {resume_operation} meaning the truck has been replaced.
- **Trucks** are always ready to receive packages and automatically replace themselves when full or when the received package doesn't fit into the truck because its size is greater than remaining space. Before being replaced, it sends a message to the conveyor belt to pause operation and, after delay of 1 to 3 seconds, another message is sent to signal the truck has been replaced and the conveyor belt can

resume operation. When a package doesn't fit, the truck sends it back to the conveyor belt to be put back in its queue. This ensures that all packages are eventually loaded into a truck, and none are lost or destroyed.

Progress Guarantee

- **Package generator** continuously creates packages, with a random size between 1 and 3, at a regular interval and sending them to a randomly chosen conveyor belt.
- **Conveyor belts** are either periodically processing their queue, sending packages to their assigned truck, receiving rejected packages from the truck when they don't fit or paused waiting for its truck to be replaced when it is full or out of space for the next package.
- **Trucks** handle incoming packages and replace themselves, with a delay of 1 to 3 seconds, when full or when out of capacity for the next package, in this case it sends the package back to the conveyor belt before replacement.
- This all ensures that all components are always active, and every package follows a logical flow of first being generated by the package generator, sent to a conveyor belt and finally loaded into a truck.

Message Passing

- All interactions between components or even to themselves occur via message passing:
    - **Generators → Conveyors**: {new_package, Package}
    - **Conveyors → Trucks**: {load_package, Package}
    - **Conveyors → Conveyors**: {process_queue}
    - **Trucks → Conveyors**: {package_rejected, Package}
    - **Trucks → Conveyors**: {pause_operation}
    - **Trucks → Conveyors**: {resume_operation}