Semaphore assignment:
The penguins

Students:

Federico Fabbri (0001025916)

Jacopo Merlo Pich (0001038025)

Daniele Simonazzi (0001058929)

Domain of the problem

We implemented the domain of the problem with 5 actions, that each penguin execute in the following order:

1. "Rest on the ice"
2. "Walk to the water"
3. "Jump in the water"
4. "Swim"
5. "Get out of the water"

Each of these actions lasts a random amount of seconds, between 0 and the values defined at the beginning of the code: BIG_SLEEP TIME ("Rest on the ice" maximum time for big penguin), SMALL_SLEEP_TIME ("Rest on the ice" maximum time for little penguins), WALK_TIME, JUMP_TIME, SWIM_TIME, EXIT_TIME.

We used 2 semaphores for little penguins (*semp1*, *semp2)*, and 1 semaphore for the big penguin (*semP*)*:*

- *semp1:* to handle action 1 of the little penguins, and are ready to start the cycle (action 2, 3, 4, 5);
- *semp2*: to handle the action 3→4 of the little penguins, so the beginning of the "Swim" action;
- *semP*: to handle all the big penguin actions that need to be synchronized with the small penguins (2→3, 4→5).

Since some little penguins may be ready to execute the "Walk to the water" action while other ones are executing the "Swim" action, we need two semaphores for the little penguins in order to avoid synchronization problems.

Little penguin

When the “Rest on the ice” action ends, the little penguin is re go and walk to the water, but it has to wait for the big penguin to go first and only then they can follow it. When it receives the signal from the big penguin, it executes both the “Walk to the water” and the “Jump in the water” actions. If it’s the last little penguin jumping in the water, it sends a signal to the big penguin, so it can jump in the water too. Once it’s in the water, it does not start the “Swim” action until the big penguin is in the water. When the “Swim” action terminates, the little penguin executes the “Go out of the water”, and the cycle starts over, from the “Rest on the ice” action and on.

Big Penguin

When the “Rest on the ice” action ends, the big penguin checks if there is any little penguin ready to walk to the water. We have two possible scenarios.

<u>Scenario 1:</u> number of ready little penguins ≠ 0. In this case, there are *N* (*n_pen*) little penguins ready to go and walk to the water. The big penguin executes the “Walk to the water” action first, then it stops and it sends *N* signals to the *N* ready little penguins, so they can execute the two actions “Walk to the water“ and “Jump in the water“. The big penguin executes the “Jump in the water” action only when it receives the signal from the last little penguin that jumped in the water. At this point, all the *N* little penguins and the big penguin are in the water together. The big penguin sends *N* signals to the little penguins, so they can execute the “Swimming” action, and it starts swimming too, so all the penguins swim together. When the “Swimming” action is completed, the big penguin waits for all the little ones to finish the “Go out of the water” action. When it receives the signal from the last little penguin that went out of the water, it executes the “Go out of the water” action too, and the cycle starts over.

<u>Scenario 2:</u> Number of ready little penguins = 0. in this case there are no little penguins ready to go and walk to the water, so the big penguin executes its entire cycle “by himself”, without little penguins. The actions 2, 3, 4 and 5 are executed consequently, and the cycle starts over.

Remarks

With this implementation of the problem domain, it often happens that some of the little penguins get ready to walk to the water when the big penguin is executing the “Walk to the water” action, so right after it checked how many little penguins were ready to go. In this case, the little penguins that were “late” have to wait until the next cycle starts.

We chose to have a shorter sleep time for the big penguin, in order to randomize as much as possible the number of ready little penguins among the cycles.

With the choices we made, we can guarantee an even number of cycles for each penguin (each penguin “swims” pretty much the same number of times, no matter how long we run the program for).

The statistics are shown at the end of the console output, via the *n_p_trips* array of integers, that keeps track of each penguin’s cycle.