Programming Assignment V

CS3331

Spring 2025

1 Santa Claus

Santa Claus has his shop up in the North Pole, where he employs r reindeer and e elves. Because of the difficulty of delivering toys across the world in one night, he prefers to spend the off-season sleeping (he needs to save energy for Christmas).

Elves toil away making toys in Santa's workshop year-round. However, every once in a while an elf will experience an issue while making toys, and the only person who can resolve these issues is Santa. One elf's problem is never serious enough to warrant waking up Santa (otherwise, he may never get any sleep). As such, Ol' Saint Nick has declared that elves may only go to wake him up in groups of three. If an elf has a problem, the elf will wait for a group of three to form before one of them (it does not matter which of the three) goes to wake up Santa from his slumber. The triumvirate submits their problems to Santa and waits until all the group's problems are answered. Then, each elf goes back to work until a new problem comes up, and Santa goes back to bed. While Santa is solving one group-of-three's problems, any other elf wishing to visit Santa must wait for those elves to return and a new group of three to form before approaching Santa.

Santa's r reindeer, on the other hand, spend the off-season on vacation, tanning on the beaches of some tropical South Pacific island. It is assumed that the reindeer don't want to leave the tropics, and therefore they stay there until the last possible moment. They might not even want to come back, but since Santa is bankrolling their year in paradise, they will begrudgingly return to the North Pole. This could explain their unprecedented present-delivery speed, since the reindeer cannot wait to get back to where it is warm. If a reindeer is the last to return from the tropics, it will wake up Santa to inform him it is time to deliver presents. Otherwise, a reindeer will wait by Santa's sleigh, waiting to be harnessed in. When Santa is awoken by the last reindeer to return, he will go attach all r reindeer to his sleigh, and the team will set off to deliver presents. After their work is done, Santa releases all r reindeer, who rush back to the South Pacific, with Santa himself going back to sleep.

If Santa wakes up to find three elves waiting at his shop's door, along with the last reindeer having come back from the tropics, Santa has decided that the elves can wait until after Christmas, as it is more important to deliver presents.

After t Christmases, Santa decides it's time to take a long vacation. At that point, Santa lays off all the elves and reindeer that work in his shop and calls it quits. This is why you have not received toys from Santa for many years!

2 Program Structure

You will design a Hoare-type monitor to simulate this problem. This program will have three classes of thread: Elves, Reindeer, and Santa. The following discusses the structure of each thread. There are a number of monitor procedures (i.e., `AskQuestion()`, `ReindeerBack()`, `FlyOff()`, `AnswerQuestion()` and `Sleep()`), each of which will be discussed below. **This is just template code, you may modify these**

as you see fit. It is encouraged that you add other public and private monitor procedures as necessary.

2.1 Elves

Each elf is a thread with the following pattern:

```
while (1) {
  Delay();                // Make toys
  monitor.AskQuestion();  // Encounter a problem
  Delay();                // Quesiton answered
}
```

`AskQuestion()` is a monitor procedure. When an elf has a question, he/she calls `AskQuestion()`, which blocks the calling elf until a gang of three is possible. When the returns, the question is answered.

2.2 Reindeer

The following shows a reindeer thread:

```
while (1) {
  Delay();                 // Tan on the beaches
  monitor.ReindeerBack();  // Report back to Santa
  Delay();
}
```

`ReindeerBack()` is a monitor procedure. Reindeer return to the North Pole one by one. When a reindeer returns, it calls `ReindeerBack()` to signal this event. Then it waits for the other reindeer to return from vacation, or, if it is the last returning reindeer, wakes up Santa. Next, all reindeer wait to be attached to Santa's sleigh. Once this step completes, all reindeer fly off to deliver toys.

2.3 Santa

Finally, the Santa thread has the following high-level behavior:

```
while (/* not retired */) {
  Delay();
  monitor.Sleep();  // Take a nap
                    // Awoken by elves or reindeer
  if (/* all reindeer are back */) {
    monitor.FlyOff();
  }
  if (/* elves have a question */) {
    monitor.AnswerQuestion();
  }
}
```

Santa sleeps first with monitor procedure `Sleep()`, which means he is blocked until three elves ask questions, or the last returning reindeer wakes him up. Delivering presents is prioritized over helping elves, and as such the reindeer are handled first. If Santa is awoken by the last reindeer, he releases all waiting reindeer. Santa then waits for all reindeer to arrive at the sleigh, at which point he attaches the reindeer to the sleigh. After the sleigh is attached, Santa flies off with the reindeer to deliver toys. On the other hand, if Santa is awoken up by the elves, he takes some time to solve the problem, and releases all three elves.

3 Other Requirements

Write a C++ program using ThreadMentor to simulate these activities. Note that you can only use one and only one monitor as specified. **Your program should not use other synchronization primitives, like semaphores.** The only exception is the use of a single mutex lock for printing to `stdout`.

3.1 Input

Input to the program is taken via command line parameters. The program is invoked as

```
./prog5 e r t
```

with the input values representing the following:

- the number of elves $e \geq 0$,
- the number of reindeer $r > 0$,
- and the number of toy deliveries $t > 0$.

Thus, `./prog5 4 9 5` means Santa's shop has 4 elves and 9 reindeer, and after delivering toys 5 times, Santa retires.

3.2 Output

The following text block contains snippets of an example output. Note that the `.....` indicates that, due to the nondeterministic nature of concurrent computing, it is possible for other print statements to occur here. Your final program should *not* print these periods.

```
Santa thread started
Santa goes back to sleep
  Reindeer 1 thread started
  Reindeer 4 thread started
.....
  Reindeer 2 returns
    Elf 2 thread started
    Elf 3 thread started
  Reindeer 5 returns
  Reindeer 3 thread started
.....
    Elf 4 thread started
    Elf 2 has a question
.....
    Elf 3 has a question
.....
    Elf 1 has a question
    Elves 2, 3, and 1 approach Santa with their question
Santa wakes up
Santa answers questions by elves 2, 3, and 1
    Elf 3 goes back to work
  Reindeer 7 returns
    Elf 4 has a question
    Elf 1 goes back to work
    Elf 2 goes back to work
  Reindeer 8 returns
```

```
Santa goes back to sleep
  The last reindeer 8 goes to wake up Santa
Santa wakes up
Santa sees all the reindeer have returned and prepares the sleigh
  Reindeer 1 is attached to the sleigh
  Reindeer 4 is attached to the sleigh
  Reindeer 6 is attached to the sleigh
.....
  Reindeer 8 is attached to the sleigh
    Elf 1 has a question
With all reindeer harnessed in, the crew sets off (1)!
The crew returns from doing deliveries
    Elves 4, 3, and 2 approach Santa with their question
  Reindeer 1 goes on vacation
  Reindeer 4 goes on vacation
.....
  Reindeer 8 goes on vacation
Santa answers questions by elves 4, 3, and 2
    Elf 3 goes back to work
    Elf 2 goes back to work
    Elf 4 goes back to work
Santa goes back to sleep
.....
  Reindeer 2 is attached to the sleigh
  Reindeer 6 is attached to the sleigh
With all reindeer harnessed in, the crew sets off (5)!
The crew returns from doing deliveries
    Elves 4, 2, and 3 approach Santa with their question
  Reindeer 7 goes on vacation
  Reindeer 4 goes on vacation
.....
  Reindeer 6 goes on vacation
Santa retires after 5 deliveries
```

Due to the nondeterministic behavior of multi-threaded programs, you will not be able to generate an output that is exactly identical to the above.

The indentation of each thread is as follows:

- All messages from the Santa thread start on column 1.
- All messages generated by reindeer threads start on column 3 (i.e., 2 leading spaces).
- All messages printed by elf threads start on column 5 (i.e., 4 leading spaces).

The output below illustrates that elves 2, 5 and 1 have problems and form a group. They wake up Santa. Then, Santa answers their questions and release them. After this, the elves go back to work and the Santa goes back to sleep.

```
    Elf 2 has a question
    Elf 5 has a question
    Elf 1 has a question
Santa answers questions by elves 2, 5, and 1
    Elf 5 goes back to work
    Elf 1 goes back to work
    Elf 2 goes back to work
```

The following illustrates the interaction between reindeer and the Santa. Reindeer return one by one. The last one, reindeer 9 here, wakes up Santa. Then, Santa prepares the sleigh. After this, the team flies off. The number in parenthesis is the number of toy deliveries.

```
  Reindeer 1 returns
  Reindeer 2 returns
.....
  Reindeer 9 returns
Santa goes back to sleep
  The last reindeer 9 goes to wake up Santa
Santa wakes up
Santa sees all the reindeer have returned and prepares the sleigh
  Reindeer 4 is attached to the sleigh
  Reindeer 5 is attached to the sleigh
.....
  Reindeer 8 is attached to the sleigh
  Reindeer 9 is attached to the sleigh
With all reindeer harnessed in, the crew sets off (3)!
The crew returns from doing deliveries
  Reindeer 4 goes on vacation
  Reindeer 5 goes on vacation
.....
  Reindeer 8 goes on vacation
  Reindeer 9 goes on vacation
```

After a specific number of toy deliveries, Santa prints out a message and stops his toy-making operation. Note that this has to be the last message printed by your program.

```
Santa retires after 5 deliveries
```

3.3 Specification

Your implementation must correctly handle the following important situations:

1. Santa can only be awoken while he is sleeping.
2. Only exactly three elves can form a group to ask Santa questions.
3. Santa cannot answer questions until he is awoken.
4. An elf will not leave the question group until Santa has answered the questions.
5. If an elf has a question, he will not return to working until it is answered.
6. When Santa is awoken by a reindeer, it must be the last reindeer to return from vacation. Only the last reindeer can wake up Santa.
7. The last reindeer will only wake up Santa while he is sleeping. If the last reindeer arrives while Santa is awake, he will wait for Santa to sleep before he wakes him up.
8. While Santa is attaching the sleigh and delivering toys, all reindeer must be present.
9. Elves will only approach Santa while he is sleeping (waking him up), or if he is awake, only after he has attended to his deliveries.

10. Santa only answers one group of elves' questions during any given period of being awake. If ≥ 3 elves have a question but Santa already has answered another group's questions, they must wait for him to go back and get some sleep before waking him up.
11. If both elves and reindeer approach him at the same time, Santa prioritizes doing deliveries first, and then answers elves questions.
12. The "Santa retires" message must be the last message printed by the system.

4 Submission Guidelines

The following sections elaborate submission requirements not directly related to the semantics of the program itself.

4.1 General Rules

1. All programs must be written in C++, programs written in other languages will receive a 0.
2. All programs must use ThreadMentor, programs that make use of non-ThreadMentor concurrency/synchronization libraries will receive a 0.
3. Your work must be submitted as a `.zip` file via canvas. This `.zip` will contain your source code, your `Makefile`, and your `README`.
4. Your `Makefile`, when passed the `noVisual` target, should produce an executable file named `prog5` using the non-visual version of ThreadMentor.
5. We will use the following approach to compile and test your program:

   ```
   make noVisual # make your program
   ./prog5 e r t # test your program
   ```

   where e, r, and t are placeholder values. This procedure may be repeated a number of times with different input configuration to see if your program behaves correctly.
6. Your program is only allocated 5 seconds to execute. If your program takes longer than this to complete, it is assumed to be in a deadlocked / non-terminating state, thus you will lose points.
7. Your implementation should fulfill the program specification as stated.

4.2 Program Style and Documentation

- At the top of each `.cpp`/`.h` file, the first comment should be a program header to identify yourself like so:

  ```
  // -----------------------------------------------------------
  // NAME : John Smith User ID: xxxxxxxx
  // DUE DATE : mm/dd/yyyy
  // PROGRAM ASSIGNMENT #
  // FILE NAME : xxxx.yyy
  // PROGRAM PURPOSE :
  // A couple of lines describing your program briefly
  // -----------------------------------------------------------
  ```

  Here, User ID is the one you use to log in to MTU systems.
- Your programs must contain a reasonable amount of concise and to-the-point comments. Do not write a novel.

- Your program should have good indentation.
- You should avoid using global variables. All of the monitor's state should be encapsulated in the monitor class itself.
- You should include a number of calls to the `Delay()` method in your program to increase its dynamic behavior.

4.3 The `README` File

A `README` file should be included with your submission. This file must either be a plaintext file named `README`, or a PDF file named `README.pdf`. Make sure you have a convincing argument for each question. While a formal proof is not expected, you should give sound reasoning for why your program will not break the specification. Arguments such as "because a monitor and conditional variables are used, the indicated situation cannot occur" will not be counted as convincing.

1. How do you guarantee that only three elves will ask questions?
2. Show that why no elf will leave before the questions are answered.
3. Show that while three elves are waiting for an answer, no other elves can cut in and ask questions.
4. How do you guarantee that Santa only answers questions while he is not sleeping?
5. Show that when Santa is awoken by a reindeer, this reindeer is the last one coming back from vacation.
6. Show that while Santa is attaching the sleigh and delivering toys, all reindeer are there. That is, they won't sneak out for vacation.
7. Show that while Santa is attaching the sleigh and delivering toys, elves will not ask questions.

Make sure to indicate clearly which question you are answering. Even if the program submitted does not satisfy one of the given specification requirements, you can still receive points for your reasoning about the safety of the system in the `README`. So, even if your program does not work very well, put effort into these answers!

4.4 Submission File Structure

Your submission should follow this file structure:

```
submission/
  thread-main.cpp   # main program implementation
  thread.h          # thread definitions
  thread.cpp        # thread implementations
  santa-monitor.h   # monitor definition
  santa-monitor.cpp # monitor implementation
  README{.pdf,.txt,}
  Makefile
```

Please do not write code in files outside the ones listed above, as you may lose points for not following the program file structure. Remember that file names in UNIX-like systems are case-sensitive, so `README.txt` is a different file from `readme.txt`!

5 Final Remarks

This, like the last program, can be incredibly difficult. You may find yourself running into issues with your program that are difficult to reason about. Thus, it behooves you to start early to give yourself time to work on any issues you may run into, as well as to seek help.

Finally, run through this quick checklist before and after you submit!

- My submission consists entirely of my own work, and I know I risk failing the entire course if I have plagiarized from any source.
- My program does not use semaphores, except for a potential mutex to protect printing to `stdout`.
- I am submitting the correct program to the correct assignment on canvas.
- My program follows the program file structure described in 4.4.
- I have included a `Makefile`.
- I have included my `README` file, and it is either a plaintext or PDF file.
- I have tested to make sure my `Makefile` works correctly, and successfully compiles the program using the command given in section 4.1.
- I have tested my program to ensure it works with several inputs, including the given example input.
- My program's output conforms to the output requirements described in section 3.2.
- My program takes input as command line parameters, as described in section 3.1.
- My program is written in C++.
- My program only uses ThreadMentor for concurrency and synchronization. I do not use pthreads or any other threading library.
- I have tested both compiling and running my program on a CS lab machine, or `colossus.it.mtu.edu`.