CSE 4/586 Project: Termination detection

There are N processes numbered 1..N connected by channels. A process can be in one of two possible states: active or idle. Initially all processes are active, and all channels are empty.

A process that is in the active state can perform the following actions:

1. send a message
2. receive a message, or
3. change state to be idle.

An idle process, on the other hand, can only perform a single action: receive a message, at which point it becomes active. (A process can read only its own state and the sequence of messages it sends and receives.)

There is a special **detector** process, process 0, that aims to detect termination of the processes 1..N. When a process becomes idle, it sends a message to the detector. So, when the detector has heard from all processes, it knows that all processes became idle. However, simply indicating a transition to idle is not enough. The global state of a distributed system consists of the states of all processes and communication channels in the system. So the processes should supply the detector information about the state of the channels as well.

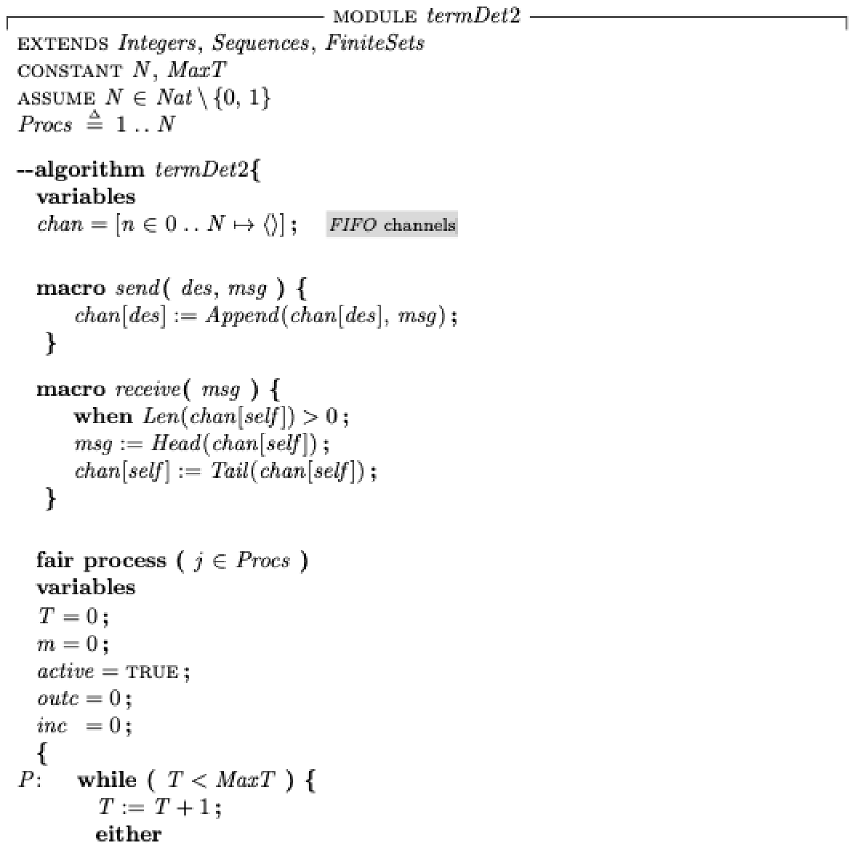
The detector should declare **done** when it detects the termination occurred, i.e., when all processes are idle and there is no message in transit to any process. The detector should provide these properties:

- safety: invariant.(**done** $\Rightarrow$ computation has terminated)
- progress: computation has terminated $\leadsto$ **done**

1 Write a PlusCal specification to model this termination detection algorithm (100 points)

I give you this template to get you started. Start by using this template, and make sure you keep the same variable names as you make the modifications instructed.

In this template, the processes do not keep track of how many messages are sent to every other process, which requires N integers at each process (a process can also send a message to itself). Instead each process only keeps total messages sent which requires 1 integer, and total messages received, using another integer.



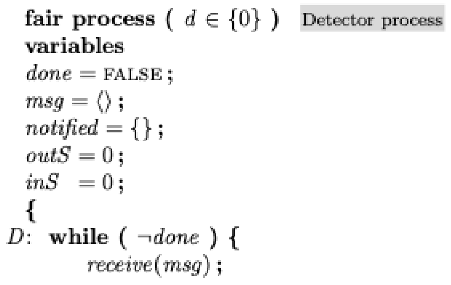
```
MODULE termDet2
EXTENDS Integers, Sequences, FiniteSets
CONSTANT N, MaxT
ASSUME N ∈ Nat \ {0, 1}
Procs ≜ 1 .. N

--algorithm termDet2{
  variables
  chan = [n ∈ 0 .. N ↦ ⟨⟩];   FIFO channels

  macro send( des, msg ) {
      chan[des] := Append(chan[des], msg);
   }

  macro receive( msg ) {
      when Len(chan[self]) > 0;
      msg := Head(chan[self]);
      chan[self] := Tail(chan[self]);
   }

  fair process ( j ∈ Procs )
  variables
  T = 0;
  m = 0;
  active = TRUE;
  outc = 0;
  inc  = 0;
   {
P:    while ( T < MaxT ) {
         T := T + 1;
         either
```



```
  fair process ( $d \in \{0\}$ )   Detector process
  variables
  $done = \text{FALSE}$ ;
  $msg = \langle\rangle$ ;
  $notified = \{\}$ ;
  $outS = 0$ ;
  $inS\ \ = 0$ ;
  {
D:  while ( $\neg done$ ) {
        $receive(msg)$ ;
```

Model the system with this setup. Use the toolkit to translate your code to TLA+ and model-check for safety and progress properties mentioned above. Use N=3 and MaxT=3. You will find that the safety property is violated. Take note of the counterexample violating the safety and describe what goes wrong. Write a report in pdf, and explain your findings/observations about how the Safety property is violated. **20 points** will be awarded to your explanation.

Now, fix the problem by making each process maintain more information. Each process should maintain N integers to keep track of how many messages are sent to every other process. Each process also maintains 1 integer to denote how many messages it received in total. Make processes communicate this information to the detector, when they go idle. Fix the detector so both safety and progress properties are achieved.

- **40 points** will be awarded for the fixed model you write and the report you write in the comments explaining how your model solves termination detection problem correctly. The grading here will be adjusted based on the correctness of the properties and your model. Having a good explanation of a wrong/incomplete property does not get full mark.
- Safety property and its correctness checking will be awarded **(20 points)**.
- Progress property and its correctness checking will be awarded **(20 points)**.

2 Submission

Your TLA+ file should be named *termDet.tla*. Your model's name should be the default name *Model*_1 (do not name your model file differently). Create a zip file from the ".tla" file and the corresponding ".toolbox" directory. **Name the zipfile as: proj.zip**

Your report needs to be named *report.pdf*. Make sure to include the names of group members in the report. A group can have at most two members, or you can choose to submit alone instead of forming a group. We will not help forming groups, and we will not accept excuse about your group member bailing on you. You can use Piazza for finding a partner to group with. Piazza has a dedicated search for teammate section.

You will use the submit command ($submit\_cse486$ or $submit\_cse586$ respectively) to submit your work. The submit command instructions are here: `http://wiki.cse.buffalo.edu/services/content/submit-script`

You can have teams upto two people. Include the names and UB-ids of the team members as the first line in the comments section.

The submission deadline is 11/15 (11:59pm EST). **No late submissions will be accepted.** An early submission at 11/10 will receive a 10% bonus points. The bonus points are decreased by 2 every passing day, and submission at deadline of 11/15 (11:59pm EST) will have zero bonus points.

3 Academic integrity policy

We have zero tolerance on cheating! A copied or plagiarized homework may lead to receiving a failing grade for the course with permanent notation on the transcript that the grade of "F" was assigned for reason of academic dishonesty. Team members are equally responsible. Consult the University Statements on Academic Integrity: `https://engineering.buffalo.edu/computer-science-engineering/information-for-students/policies/academic-integrity.html`

Students who do share their work with others are as responsible for academic dishonesty as the student receiving the material. Students are not to show work to other students, in class or outside the class. Students are responsible for the security of their work and should ensure that printed copies are not left in accessible places, and that file/directory permissions are set to be unreadable to others.

Excuses such as "I was not sure" or "I did not know" will not be accepted. Make sure you go through these questions, and check your answers to these by reading the links on academic integrity.