Computer Science 315 Laboratory 7
The Producer-Consumer Problem

Due: 11:59 pm on Monday March 27, 2023.

The Dining Philosophers is a good example of deadlock situations. In this lab, you will be exploring this problem and also competing in a game competition with the student with the best agent getting a bonus mark.

PART 1 – The problem

Some philosophers, for now let's say 5, have gathered at a dining table to eat and think about things. Between thinking sessions, philosophers need to use the community chopsticks to grab food for their dish. The problem is, there are only 5 serving chopsticks between the 5 philosophers so each philosopher needs to share a chopstick with the person on their left and the person on their right.

Exercise 1 (3 marks):

a) With the restrictions above, how many philosophers can be eating at a given time?
b) What if there were 10 philosophers (and 10 chopsticks)? How many can eat at once?
c) What if there were 3 philosophers (and 3 chopsticks)? How many can eat at once?

PART 2 – The game

In Moodle there is the source code to an incomplete dining philosophers' game. The rules to this game are as follows:

Every philosopher wants to think as much as possible with thinking earning a point.

Thinking takes energy. If no energy, can't properly think so no points.

Philosophers can only grab one chopstick at a time or can drop whatever chopsticks they hold.

Once a philosopher has both chopsticks, they can eat.

Eating gives the philosopher 3 points of energy but they can only have a maximum of 5 energy.

If there is a potential deadlock, philosophers will be given a warning and if still in a potential deadlock the next round then the game will be over and the winner will be the philosopher with the highest score.

Exercise 2 (4 marks):

Take a look at the provided code. You will notice that all the logic for the game is contained within struct dining. This structure is created and used to control the game. One of the pieces of information

Right now there is only a placeholder method called isDeadlock(…) which is suppose to check if a deadlock is happening. This requires checking two things and if this is present, then to set the deadlock warning for the philosophers to give them a chance to resolve the situation. If they do not resolve the situation, return a non-zero value (TRUE) to indicate that the game is over.

Write the deadlock detection code.

PART 3 – Avoidance

There are several ways of avoiding a deadlock. Many require that processes follow a set of rules (which may not always be enforceable). Take a look at the AI_Basic function to get an idea of how to write an agent. The commands available to the agent are:

THINK which earns points at cost of energy. If no energy, no points are earned. Thinking while holding a chopstick does not earn points.

GRAB_LEFT will attempt to grab the chopstick to the left of the player

GRAB_RIGHT will attempt to grab the chopstick to the right of the player

EAT will eat then drop chopsticks gaining energy, but only if player has BOTH chopsticks.

DROP will drop the chopstick(s) that the player is holding.

Exercise 3 (4 marks):

Using the information provided in the struct dining structure provided, create a function that implements an agent that will attempt to avoid a deadlock situation from happening.

The agent being used for the game can be modified by just changing the function pointer for handling the AI. Look at the main section of the program and replace one of the AI_Basic agents with your own agents.

PART 4 – The competition

The goal of the game is to get as much thinking done as possible before the game is over. There are many different strategies that can be applied so think about ways you would compete to get the highest score in this game. Some ideas: order of the chopsticks, other player eating habits, deadlocks may be an advantage, energy timing.

Exercise 4 (5 marks):

Write an agent with the goal of getting the highest possible score. Your agent will be pitted against the agents from your class-mates to decide what your score for this section is. Several games will be held and the winner of the most games will get a bonus point if they are present in class during the competition.

Submission Instructions

Submit part 1-4 on Moodle by due date. The answers to Part 1 can be a separate text file or comments at the top of the submitted dining.c file. You may continue to work on part 4 until the in-class competition on March 31st.