Programming Assignment 2

COP3502 Computer Science I – Spring 2021

Overview

This assignment is intended to teach you to use a circular doubly-linked list and its queue.

Details

Failfish are small monsters that eat only other Failfish. This makes them ecologically and practically useless, causing many small monster scholars to wonder how they ever came to exist in the first place.

They *are* delicious, which is a plus, but since they eat each other, they're almost impossible to farm. That doesn't keep people from trying. It keeps them from succeeding, but not from trying.

You have been hired by an ambitious small monster farmer to record their attempts to farm Failfish.

The farmer has:

- Ten Failfish ponds arranged one after the other.
  - Each pond has a name, and also a sequence number from 1 to 10.
- G groups of Failfish that they've captured. Groups of Failfish are called Failgroups. They were called Failschools once, but small monster scholars determined that Failfish are unable to learn.
  - Each Failgroup g_i has n_i Failfish in it, with $n_i \geq 2$. Note that i is the **pond number** – the Failfish don't have to be in ponds $1..G$.
  - Each Failfish in a Failgroup is branded with a sequence number, from 1 to n_i.
    - (Modern small monster farming methods being humane, the branding is done with a particularly waterproof marker. No Failfish are harmed during modern Failfish farming. (Except when they get eaten.))

The Meal, First Course:

Failfish in a Failgroup form a Failcircle when preparing to eat each other. For a group of seven Failfish, the Failcircle looks like Figure 1.

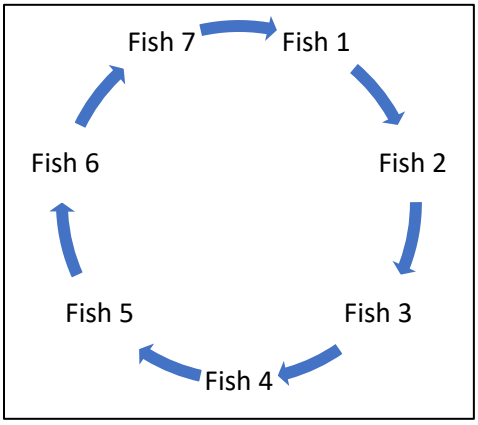

Figure 1: A typical Failcircle.

Once a Failcircle is formed, phase 1 of the dining begins. It's controlled by three values: e_i, the eating counter; th_i, the minimum Failgroup threshold; and n_i, the number of Failfish.

- The Failfish begin counting with the smallest-numbered Failfish in the Failcircle.
- The Failfish skip the first $e_i - 1$ Failfish, and eat the e_ith Failfish clockwise of the starting Failfish.
- The Failfish then skip the *next* $e_i - 1$ fish, and eat the e_ith Failfish clockwise of the last-eaten Failfish.
- This process continues until the Failcircle's population drops to th_i, the minimum Failgroup threshold.

Each Failgroup dines independently according to this algorithm. No one is quite sure why.

The fish will continue to eat each other until the population of the group is reduced to a threshold value th_i. Uneaten fish at this point continue to the second course.

The Meal, Second Course:

At this point the Failcircles unwind into Failqueues, with the smallest-numbered Failfish from each Failcircle at the head of that Failcircle's Failqueue.

- The Failfish converge on and consume the highest-numbered Failfish *of those at the heads of the queues*.
- In case of a tie, the Failfish at the head of the smallest-numbered Failgroup's queue is eaten.
- This continues until there is only one Failfish remaining.
- You are to report which Failfish this is, based on the theory that this information will allow the farmer to breed Failfish that are less likely to be eaten.
    - (It will not.)

Input and Output

Read input from **cop3502-as2-input.txt**. Write output to **cop3502-as2-output-<yourlname>-<yourfname>.txt**. For example, my output file will be named **cop3502-as2-output-gerber-matthew.txt**.

There are blank lines in the sample inputs to make them more readable. They may or may not be present in the actual inputs; you should completely ignore blank lines.

The first line of the input file contains the number of Failgroups. Each line after that contains, separated by spaces:

- The pond number g_i
- A name for the pond
- The value n_i $(2 \leq n_i \leq 10{,}000)$
- The value e_i $(1 \leq e_i < n_i)$
- The value th_i $(1 \leq th_i < n_i)$

The output format is shown in the sample output, on the next page.

Sample Input

```
5

4 mishap 7 3 3
6 futility 5 2 3
1 failure 10 3 2
7 sisyphean 9 2 4
3 doomed 8 2 1
```

Sample Output (both columns are part of the output)

```
Initial Pond Status
1 failure 1 2 3 4 5 6 7 8 9 10
3 doomed 1 2 3 4 5 6 7 8
4 mishap 1 2 3 4 5 6 7
6 futility 1 2 3 4 5
7 sisyphean 1 2 3 4 5 6 7 8 9

First Course

Pond 1: failure
Failfish 3 eaten
Failfish 6 eaten
Failfish 9 eaten
Failfish 2 eaten
Failfish 7 eaten
Failfish 1 eaten
Failfish 8 eaten
Failfish 5 eaten

Pond 3: doomed
Failfish 2 eaten
Failfish 4 eaten
Failfish 6 eaten
Failfish 8 eaten
Failfish 3 eaten
Failfish 7 eaten
Failfish 5 eaten

Pond 4: mishap
Failfish 3 eaten
Failfish 6 eaten
Failfish 2 eaten
Failfish 7 eaten

Pond 6: futility
Failfish 2 eaten
Failfish 4 eaten

Pond 7: sisyphean
Failfish 2 eaten
Failfish 4 eaten
Failfish 6 eaten
Failfish 8 eaten
Failfish 1 eaten

End of Course Pond Status
1 failure 4 10
3 doomed 1
4 mishap 1 4 5
6 futility 1 3 5
7 sisyphean 3 5 7 9

Second Course

Eaten: Failfish 4 from pond 1
Eaten: Failfish 10 from pond 1
Eaten: Failfish 3 from pond 7
Eaten: Failfish 5 from pond 7
Eaten: Failfish 7 from pond 7
Eaten: Failfish 9 from pond 7
Eaten: Failfish 1 from pond 3
Eaten: Failfish 1 from pond 4
Eaten: Failfish 4 from pond 4
Eaten: Failfish 5 from pond 4
Eaten: Failfish 1 from pond 6
Eaten: Failfish 3 from pond 6

Failfish 5 from pond 6 remains
```

Implementation Requirements

- You must use the circular doubly linked list, and a queue based on it.
- Declare an appropriate node structure to store a Failfish's sequence number as the data value.
- Declare an appropriate list structure to include the pond name, n_i, e_i and th_i.
- You'll need enqueue(), dequeue(), peek(), and is_empty().
- Required functions and their prototypes:
    - failfish *create_failfish(int sequence_number);
    - failfish_queue *create_failfish_queue(char *pondname, int n, int e, int th);
    - void print_failfish_queue(failfish_queue *q);
- Use the leak detector from Assignment 1.
- I expect to see constructors and destructors for each of the structure types, with appropriate parameters.
- You do not need to comment line by line, but comment every function and every "paragraph" of code.
- You don't have to hold to any particular indentation standard, but you must indent and you must do so consistently within your own code.
- You may not use global variables.