TIC1001—Introduction to Computing and Programming I
National University of Singapore
School of Continuing and Lifelong Education

Practical Examination 2

16 Nov 2019

Time allowed: 2 hours

Instructions (please read carefully):

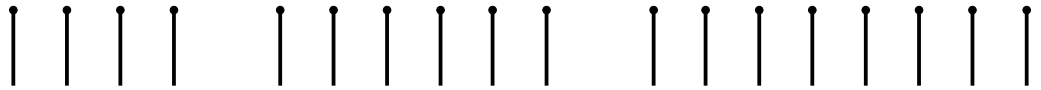
1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **two** questions. The time allowed for solving this test is **2 hours**.
3. The maximum score of this test is **20 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the templates `pe2-template.cpp`to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness locally on VS Code and not depend only on the provided test cases. Do ensure that you pasted your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `pe2-<student no>.c` where `<student no>` is your student number and leave the file on the Desktop. If your file is not named correctly, we will choose any .cpp file found at random.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
8. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.
9. Marks will also be awarded for **good coding style** like suitable variable names, proper indentations, etc., in addition to the correctness of the program.

ALL THE BEST!

Question 1: Game of Nim [20 marks]

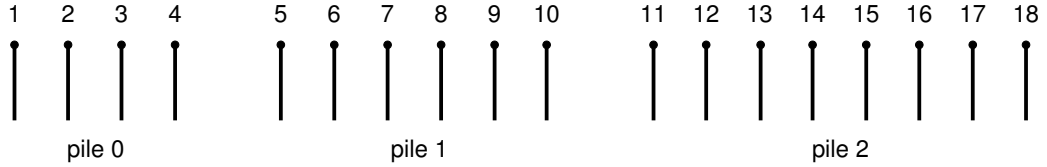
Nim is a mathematical game of strategy in which two players take turns removing objects from distinct heaps or piles. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap/pile. Depending on the version being played, the goal of the game is either to avoid taking the last object, or to take the last object. *- Source: Wikipedia*

In our version of Nim, we will be playing using matchsticks. The matchsticks are separated into piles, and are lined up within each pile, like below:



The above game has three piles of 4, 6, and 8 sticks. The state of the game is represented as a `vector` of `int`, where each element represents the number of sticks in a pile. For example, the game state illustrated above would be a vector `{4, 6, 8}`.

A. **[Warm up]** Suppose we number the matchsticks from 1 to n, starting with the first matchstick in the first pile, all the way across the piles to the last matchstick, as shown:



Implement the function

```
int pile(vector<int> game, int n)
```

that takes as input a game state and returns the pile number of the nth matchstick. You may assume that the first pile is number 0 and that n will not be more than the number of matchsticks.

[5 marks]

Sample tests:

Test function	Output
`pile({4, 6, 8}, 5);`	`1`
`pile({4, 6, 8}, 16);`	`2`

B. The game of Nim is played by players removing matchsticks from a single pile. Implement the function

```
void remove(vector<int> &game, int pile, int n
```

which takes in a game state and updates it by removing n matchsticks from the given pile. Note that when a pile is completely removed, it should also be removed from the game state, i.e. there should not be any pile with 0 matchsticks.

You may assume that the inputs are valid, i.e. n is not more than the size of the specified pile.

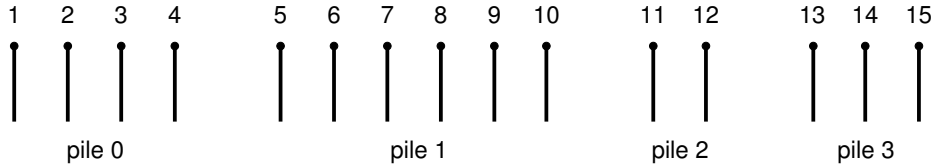
[7 marks]

Sample tests: Initial content of `game = {4, 6, 8}`

Test function	Content of `game`	Remarks
`remove(game, 0, 2);`	`{2, 6, 8}`	2 sticks removed from pile 0
`remove(game, 1, 2);`	`{2, 4, 8}`	2 sticks removed from pile 1
`remove(game, 2, 2);`	`{2, 4, 6}`	2 sticks removed from pile 2
`remove(game, 1, 4);`	`{2, 6}`	4 sticks removed from pile 1, pile is removed

C. Consider this variation in the rules: when some matchsticks in the middle of a pile are removed, the pile might be split into two separate piles.

For example, suppose the game state is `{4, 6, 8}` as shown in part A, and matchsticks 13 through 15 are removed. The new game state will look like this:



Essentially, pile 2 was split into a new pile 3, giving `{4, 6, 2, 2}`. As before, if a pile is completely removed, then it will also be removed in the game state, e.g. remove matchsticks 11 through 12 now will result in `{2, 6, 3}`.

Implement the function:

```
void split(vector<int> &game, int first, int last)
```

which takes in a game state, and the first through last matchsticks to be removed in a turn. Similar to part A, the matchsticks are consecutively numbered 1 to n across the piles.

If the matchsticks to be removed stretches across piles, then the move is illegal and nothing happens. Otherwise, if it is a legal move, the function will update the game state to a new state having the stated matchsticks removed. [8 marks]

Sample tests: Initial content of `game = {4, 6, 8}`

Test function	Content of `game`	Remarks
`split(game, 13, 15);`	`{4, 6, 2, 3}`	Remove matchsticks 13–15. Pile 2 is split.
`split(game, 11, 12);`	`{4, 6, 3}`	Remove matchsticks 11–12. Pile 2 is removed.
`split(game, 5, 7);`	`{4, 3, 3}`	Remove matchsticks 5–7. No new pile created.
`split(game, 3, 6);`	`{4, 3, 3}`	Remove matchsticks 3–6. Invalid mov4.

— E N D O F P A P E R —