COP-4338 Programming III, Summer 2018

Programming Assignment 2:
Introduction to C & Functions in C

Due Date: June 7 at 11:55 PM

This assignment has two parts: 1) Implementing a recursive function that calculates the remainder of dividing a very large integer by three and 2) simulating a queue data structure using two stacks.

1 Remainder of Dividing by Three

The easiest way to find the remainder of a given integer like 235 when dividing by three is to calculate the remainder of $2+3+5$ instead. Use this idea to implement a *recursive* function that calculates the remainder of any given non-negative integer when dividing by three.

Your function must obey the following signature:

int nMOD3 (int n[ ], int size)

where array n specifies the input integer by storing each digit in one cell and *size* determines how many digits the input integer has. For example, *nMOD3 ({2, 3, 5}, 3)* is a valid function call. The remainder of n when dividing by three has to be returned by the nMOD3 function as its output.

2 Constructing a Queue

Queue is a data collection in which the entities in the collection are kept in order and the operations on the collection are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue. This makes the queue a First-In-First-Out (FIFO) data structure. As we mentioned in the class, stack is a similar data collection that adds/removes entities in a First-In-Last-Out (FILO) manner.

In this assignment, you need to construct a *queue of integers* by implementing its enqueue and dequeue operations using two stacks in the following way (implementation of a stack and its push & pop operations in C can be found in the textbook and in slides).

- void enqueue (int entity): The enqueue operation can be done by simply pushing the entity in the first stack.

- int dequeue(): In order to implement dequeue operation, you need to first transfer *all* of the entities stored in the first stack to the second one using a sequence of alternating pop and push operations on the first and second stacks respectively (i.e. popping an integer from the first stack and pushing it back to the second one). After emptying the first stack, you simply pop from the second stack to complete the dequeue operation.

3 Submissions

You need to submit a *.zip* file compressing the following folder:

- A folder named "1" containing all the C source file(s) related to the first part of the assignment (*.c* files)
- A folder named "2" containing all the C source file(s) related to the second part of the assignment (*.c* files)