PDS 2710: Homework Assignment #5

Due on Sunday, September 18, 2016

Problem 1

Stack

You are supposed to implement Stack ADT using both Linked List and Arrays.

Stack implementation using linked list

Implement **stack** and operations on it using **Linked-List** implementation of previous assignment. Functions corresponding to the operations on stack and the structure of the stack are declared in file **stack.h** and are supposed to be implemented in **stack.c**

Stack implementation using array

Implement **stack** ADT and operations on it using **Array**. Functions and the structure of the stack are declared in the file **stack_a.h** and are supposed to be implemented in **stack_a.c**. Size of the array is defined to be constant value 1000

```
1. stack* stack_new()
2. void stack_push(stack*, int)
3. int stack_pop(stack*)
4. bool stack_is_empty(stack*)
5. bool stack_is_full(stack*)
6. int stack_size(stack*)
7. void stack_print(stack*)
```

Note: function #5 must only be implemented in array-based implementation as it doesn't make sense to implement in list-based implementation. To check the correctness of your implementation use driver programs **q1_stack.c** and **q1_stack_a.c** provided with resources. As already instructed during previous lab, please check the correctness of each of your functions' implementations before using driver programs. Please refer to your lecture-notes or text-book to understand the structure of the stacks and operations them using both approaches.

Convert an Infix Expression to Postfix Expression

Given an infix expression, convert it into postfix Expression. Reading on Postfix, Prefix and Infix notations
Example:
Input: A / (B * (C + D))
ouput: A B C D + * /

Note: Assume that that each term, including operators, in the input expression will be separated by space and hence output terms must also be separated by spaces.

Evaluate a Postfix Expression

Evalute a Postfix expression over integer terms.
Example:
Input: 6 2 3 + - 3 8 2 / + * 2 * 3 +
ouput = 17
Note: Assume that each terms in Postfix expression, including operators, are separated by space and allowable operators in the expression are : **+ - * /**

Problem 2

Queue

Similarly, You are supposed to implement Queue ADT using both Linked List and Arrays.

Linked List Implementation of Queue

Use your linked list implementation to implement the following functions in **queue.c**, declared in **queue.h**

Queue implementation using array

Use array to implement the **queue** ADT and following operations on it in **queue_a.c**, declared in **queue_a.h**. Size of the array has been defined to be constant 1000.

```
1. queue* queue_new()
2. void enqueue(queue* , int)
3. int dequeue(queue*)
4. bool queue_is_empty(queue*)
5. bool queue_is_full(queue*)
6. int queue_size(queue*)
7. void queue_print(queue*)
```

Note: Function #5 must only be implemented in array-based implementation of queue. Driver programs **q2_queue.c** and **q2_queue_a.c** have been included in resources and can be used to check your implementation. Refer to text-book/lecture-notes to understand the structure of the queues and operations on them

After you have written all the functions required to manipulate the queue in queue.c you are supposed to implement two programs using your implementation of the queue

Simulate Stack Using Queue

Use your queue implementation to simulate the behavior of a stack. You are supposed to implement all the operations on stacks using this approach. Simulate the stack means you are supposed to write functions on the queue but externally it should behaves like they have been performed on the stack.
For example when I call method
push(A , 12) where A is a queue, It will look like
12

Again when I push two more elements
push(A , 6)
push(A , 3)
the function should return a queue which should look like a stack. i.e First in Last out
3
6
12
So now when I call function
pop(A) on the queue **A** , the function should pop the topmost element and return a queue which looks like a stack.i.e
6
12

Simulate Queue Using Stack

Use your stack implementations to simulate the behavior of a queue. Implement all the queue-operations using your stack-implementation.
Simulate the queue means you are supposed to write functions on the stack but externally it should behave like they have been performed on the queue.
For example when I call method
enqueue(A , 12) where A is a stack, It should look like
12
Again when I push two more elements
enqueue(A , 6)
enqueue(A , 3)
the function should return a stack which should look like a queue. i.e First in First out
12 6 3
So now when I call function
dequeue(A) on the stack **A** , the function should pop the front most element and return a stack which looks like a queue by giving following output
6 3

Deliverables and Due date

You only need to submit the *.h* and *.c* files by **11:59 pm, Sunday, 18th September, 2016**.
README File that explains how to compile and run the program; whether your programs works correctly or whether there are any known bugs/errors in your program.