Helwan University
Faculty of
Computers and Information

Review Problems

Department of Computer Science
Data Structures
Spring 2007

For all of the following problems, when it is stated "in the user level" you should assume that you only have the object code of the data structure along with the interface given in the header file. On the other hand, When it is stated that "in the implementation level" you should assume that you have the source code file of the data structure and you should write the desired piece of code within that source code file.

Also assume that the stack and queue structures are implemented as linked stack and linked queue respectively.

1. Write a function that returns the first element in a stack. (implementation level)
2. Write a function that returns the last element in a queue. (implementation level)
3. write a function that returns a copy from the last element in a stack. (implementation level)
4. write a function that returns a copy from the first element in a queue. (implementation level)
5. Write a function to destroy a stack. (implementation level)
6. Write a function to destroy a queue (implementation level)
7. Write a function to copy a stack to another. (implementation level)
8. Write a function to copy a queue to another. (implementation level)
9. Write a function to return the size of a stack (implementation level)
10. Write a function to return the size of a queue (implementation level)
11. Repeat all of the problems above in the implementation level but for array-implemented structures.
12. Rewrite all of the problems above in the user level. Does it make any difference to write the functions in the user level with the structures being implemented as array-based?
13. Write a function to print on the screen the contents of a stack without changing the stack (user level). Why this function should not be written in the implementation level?
14. The previous function can be written more efficiently and professionally as follows.
    (a) Write the general function `TraverseStack` in the implementation level. This function takes, as arguments, the stack and a pointer, `pvisit`, to another function. `TraverseStack` passes `*pvisit` over every element of the stack.
    (b) The function, whose pointer is `pvisit`, is written in the application level to suite the type definition of the element.

    In such a way, each level did not assume knowledge of the other level.
15. In some applications, e.g., simulating the memory management module in the operating systems, it is required to have a fixed size structure that allows adding elements from both ends; every end is viewed as a separate stack. In such a structure, the maximum size of each stack is the size of the whole structure. However, there is another restriction; i.e., the total size of the two stacks cannot exceed the size of the whole structure.
    (a) Write the type definition of this structure.
    (b) Write the following basic functions for this structure: `CreateJoinedStack`, `PushBottom` that pushes from the logical lower end of the structure, `PushUp` that pushes from the logical upper end of the structure, `StackSize`, `StackEmpty`, `StackFull`, and `DestroyStack`.
    (c) Do we need linked implementation for this structure? Why?

    Hint: You should use appropriate names for types and variables. You should write the pre- and post-conditions of every function of the above.