1. Identify some key characteristics of stacks or stack operations.

A stack has a Last In, First Out (LIFO) structure. This means that new items are added only to the “top” of the stack, and items are removed only from the “top” of the stack (stack of plates in cafateria analogy).

Stacks are commonly implemented via arrays and linked lists.

There is no limit on the number or nature of items in the stack.

2. What methods are *required* to implement a stack?

Push – This method inserts an item into the top of the stack.
Pop – This method removes an item from the top of the stack.
A constructor – To initialize the stack

3. What methods might be helpful in managing a stack, but are not required? You might consider methods that could be implemented as a combination of methods identified in the last question.

Peek - This method will return what is on the top of the stack without actually removing it. Could be implemented as a combination of pop and push.
IsEmpty – This method checks to see if the stack is empty.
Clear – This method deletes all items from the stack, effectively “resetting” the stack.
Copy – Replicate/copy items in the stack.
Size – Return size (number of items) of the stack.

4. Name an example (or two) of an application for which stacks would be helpful, and explain why. Consider real-life applications, not just classic algorithms.

Evaluating expressions. Stacks provide a way to move step-wise through an expression and effectively evaluate it.

Backtracking. A stack can store history so that if one so desired, one could simply pop an item or items (e.g. previous event) from the stack to see what actions were previously taken. Could be useful in things such as undo in a word processor, or the back button on a web browser.

5. Name an example (or two) of an application for which stacks would NOT be helpful, and explain why. Consider real-life applications, not just classic algorithms.

It was mentioned in lecture that if resources are not available for a recursive function, the operating system will place the request for resources in a queue, which is FIFO. A stack would not be useful for this because, being LIFO, would possibly delay (possibly indefinitely) resource requests. The earlier a request was made, the longer it would take for it to be granted if a stack was used to hold the requests.

6. Have you used prefix or postfix notation in the past? In what context? Did you find it useful?

I can not think of any occasion where I have used prefix or postfix notation in the past.

7. What are some pros and cons of choosing to implement a stack using an array versus a list implementation?

Pro
- Array is very simple to implement
- Random access

Con
- Array is (in general) static. A size is allocated upon creation which cannot be exceeded.