Answer Key

1a) Answer A is false. Only classes that implements the Iterable<T> interface can be iterated over using the enhanced for loop.
1b) Answer: [1,2,3]
1c) “false” If hello was a string literal “hello”, and not a String Object, then the answer would be true.
1d)Base Derived Derived
1e) hasNext() is checking that the getNext is null from the current pointer instead of checking that the current pointer itself is null.
1f) bottom: 6 2 7 1
1g) 1 4 1 2 1 3 6

2a) O(n)
2b) O(1)
2c) O(n)
2d) O(1)
2e) O(n^2)

3)

```
public void push (T element) throws StackOverflowException {
                if (isFull())
                        throw new StackOverflowException (“push to full stack”);

                // move stuff over
                for (int i = size; i > 0; i--){
                        stack[i] = stack[i-1];
                }
                stack[0] = element;
                size++;
        }

public T pop( ) throws StackUnderflowException{
                if (isEmpty( ))
                        throw new StackUnderflowException(“pop empty stack”); {

                T ret = stack[0];

                 for (int i = 0; i < (size-1); i++)
                        stack[i] = stack[i+1];
                }

                size --;

                return ret;
        }
```

4)

```
public class LinkedStack<T> implements StackInterface<T> {

    LLNode<T> head;
    int size = 0;

    public LinkedStack(){
        head = null;  // not necessary
    }

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return (head == null);
    }

    public T pop() throws StackUnderflowException {
        if (head == null)
            throw new StackUnderflowException();
        T ret = head.getData();
        head = head.getNext();
        size--;
        return ret;
    }
```

```
public void push(T elem) {
        LLNode<T> newNode = new LLNode<T>(elem);

        newNode.setNext(head);
        head = newNode;

        size++;
}
```