1. (20%) Write a method that will reverse the elements of a Queue using a Stack. Your method should take a single parameter of type Queue, and this queue contains elements of type Object. You do NOT have to write a Stack class or a Queue class, assume that you have the Stack and Queue classes with all the necessary methods.

2a.(10%) Draw a (single) binary tree T such that

- Each internal node of T stores a single character
- A preorder traversal of T yields EXAMFUN
- An inorder traversal of T yields MAFXUEN

2b.(10%) Complete the following preorderToString method that will return the preorder traversal of the binary tree as a sequence of strings each separated by a space.

```java
public String preorderToString() {
    StringBuilder stb = new StringBuilder();
    preorderToString(stb, root);
    return stb.toString();
}

private void preorderToString(StringBuilder stb, Node<E> root) {
}
```

3.(20%) Consider the double linked list below with 3 nodes (which are referred to as node1, node2 and node3).

Given the Node class and a linked list with Node Head:

a. How would you refer to the element in the second Node of the above linked list using Head?
b. Write the code that will properly **add** an new Node with element called `newElement` **after the first** Node of the above linked list.

c. Write the code that will properly **add** an new Node with element called `newElement` **before the first** Node of the above linked list.

d. Using Head, write the code that will properly **delete the second Node of the above linked list** without destroying the list.

4. (20%) Consider the BinarySearchTree class partially given below, complete the method **add**.

```java
public class BinarySearchTree<E extends Comparable<E>> extends BinaryTree<E> implements SearchTree<E> {
    protected boolean addReturn;
    protected E deleteReturn;

    public boolean add(E item) {
        root = add(root, item);
        return addReturn;
    }

    private Node<E> add(Node<E> localRoot, E item) {
        ...
    }
}
```

5. (20%) The goal of your assignment 2 was to implement a MyPolynomial class that represents single-variable polynomials (with nonnegative exponents) by using a CIRCULAR doubly linked list. Each term in the polynomial can be stored in one node containing its coefficient and exponent. The nodes are sorted in decreasing order of exponents. For example, a polynomial \( P(x) = 10x^{100} + 5x^{14} + 1 \) can be represented by the linked list shown in the figure below.

In addition to the appropriate constructors, include the following methods in MyPolynomial:

- public int degree(): returns the degree of the polynomial.
• public String toString() : returns a String representation in the form
  \(10x^{100} + 5x^{14} + 1\)
• public MyPolynomial(MyPolynomial) : the copy constructor.
• public int getCoef(int exp): returns the coefficient corresponding to the given exponent.
• public void setCoef(int coef, int exp): sets the coefficient of the given exponent. If the term with the given exponent does not exist, it should be added to the polynomial (if coef!= 0).
• public boolean equals(MyPolynomial)
• public MyPolynomial add(MyPolynomial): adds this polynomial and the parameter polynomial, returns the resulting polynomial.
• public MyPolynomial subtract(MyPolynomial)
• public MyPolynomial multiply( MyPolynomial)
• public int evaluate(int x): evaluates the polynomial by plugging in the given value for x, and returns the result.

Write a driver to test MyPolynomial class and give a sample output of your driver.