- You can use either your own or our code from the lab sections for the List implementations.
- **This is a pair (group project) assignment**. Students form groups of two people. Both of them are responsible for the whole project.
- Do not share your code with another group or individual. Grade will be 1 for all parties.
- Do not copy-paste online codes.
- You can discuss with people around, but you cannot get help on code, or blocks of code from anyone, including relatives.
- You are only allowed to use your lecture/lab notes. Violating this rule will result in 1/100 as Project #1 grade.
- You must demonstrate your project on declared dates. Otherwise, your project grade will be 2/100.
- Both partners must be available during the project demonstration time. One (random) person from the pair (group) will be asked questions. The group grade depends on the individual's answer. It is not possible to select or change the person called for demo. It is not possible to help the person in the demo. It is not possible to explain an unanswered question with excuses such as "My friend coded these lines, I do not know them".
- If demo student fails to answer half or more than half of the questions, your project grade will be 1.
- Late submissions will not be accepted!

Problem 1 Loose the letter and table

We would like to implement two polynomials using liked lists (one way) and apply the following tasks on them:

- Add two polynomials
- Multiply two polynomials
- Print a polynomial
- Remove a node with a given power from polynomial

So, your job is to implement a separate method in Java for the tasks mentioned above.

Note: you must use the class implementation given below for the linked list and cannot use the already implemented classes for linked lists.

```
public class Node{
int data;
Node next;
public Node(int data){
this.data = data;
next= null;
}
```

```
public class MyLinkedList{
Node first, last;
public MyLinkedList(){
first= null;
last= null;
}
```

Note: Show a menu to the user and ask her to choose which one of the tasks mentioned above she wants to do? As the first task, before showing the menu, ask her to enter some coefficients together with powers for the polynomials.

Sample run:

Enter the coefficient and power of the first polynomial. Type 0 at the end:

2 5

3 3

1 -4

0

Enter the coefficient and power of the second polynomial. Type 0 at the end:

-3 5

1 6

3 0

0

The entered polynomials are:

2 x^5 + 3 x^3 + x-4

x^6 - 3 x^5 + 3

Which task you want to do?

1. **Add**
2. **Multiply**
3. **Print**
4. **Delete**

5- exit

Please enter a digit (1-5):

1

Adding two polynomials:

x^6 - x^5+3 x^3 + x - 1

Please enter a digit (1-5):

2

Multiplying two polynomials:

2x^11 - 6x^10 + 3x^9 – 9x^8 + x^7 – 7x^6 + 18x^5 +9x^3 + 3x -12

Please enter a digit (1-5):

4

Which power you want to be delated from both polynomials:

5

The elements having this power has been delated.

Please enter a digit (1-5):

4

Which power you want to be delated from both polynomials:

4

This power does not exist in any polynomial.

Please enter a digit (1-5):

3

The polynomials are:

3 x^3 + x^-4

x^6 + 3

Please enter a digit (1-5):

5

Exiting the program ….

Problem 2: Queue

We have a stack of strings and we can access only its top most element. We can use only another temporary stack. Similar to problem 1, show a menu to the user and ask her to choose one or them:

Push: can push several strings to the stack

Pop: pops only one element from stack

Print : prints the content of stack.

Ispalindrome: a stack is palindrome, if elements around the middle of stack are the same, or is not palindrome, other.

Delete middle: if the length of the stack is odd, it removes the middle element from the stack but if the length is even, it only prints appropriate message without removing anything.

Important Note: whenever necessary, you can pop values from the top of the main stack and use the temporary stack to store those values. But after doing your task, you should return those values back to the main stack. Only one temporary stack can be used, not anything else.

Note: you can use only the following Stack implementation, but not the already implemented stacks in Java.

```
public class StackArray {
   int data[]; // array to hold items
   int top; // the top most item index
   public StackArray(int N) {
      data = new int[N];
      top = -1;
   }
   public void push(int element) {
      if (!isFull()) {
         top++;
         data[top] = element;
      }
   }
```

```
public int pop() {
    if (isEmpty()) {
        throw new java.util.NoSuchElementException();
    } else {
        top--;
        return data[top + 1];
    }
}
public boolean isFull() {
    if (top == data.length - 1) {
        return true;
    }
    return false;
}
public boolean isEmpty() {
    if (top == - 1) {
        return true;
    }
    return false;
}
```

Sample run:

Which task you want to do?

1- **push**
2- **pop**
3- **Print**
4- **Delete middle**
5- **isPalindrome**
6- **exit**

Please enter a digit (1-5):

1

How may pushes you want to do?

5

Enter 5 strings:

Araz

Ehsan

Sara

Nazli

Faith

Please enter a digit (1-5):

3

The content of stack is:

Faith, Nazli, Sara, Ehsan, Araz

Please enter a digit (1-5):

5

The stack is not palindrome.

Please enter a digit (1-5):

4

The middle element has been deleted

Please enter a digit (1-5):

3

Faith, Nazli, Ehsan, Araz

Please enter a digit (1-5):

4

There is no middle element as the size of stack is even.

Please enter a digit (1-5):

2

Fatih

Please enter a digit (1-5):

3

Nazli, Ehsan, Araz

Please enter a digit (1-5):

1

How may pushes you want to do?

2

Enter 2 strings:

Ehsan

Araz

Please enter a digit (1-5):

5

The stack is palindrome.

Please enter a digit (1-5):

6

Exiting …

Good luck

Rahim Dehkharghani, Comp2112