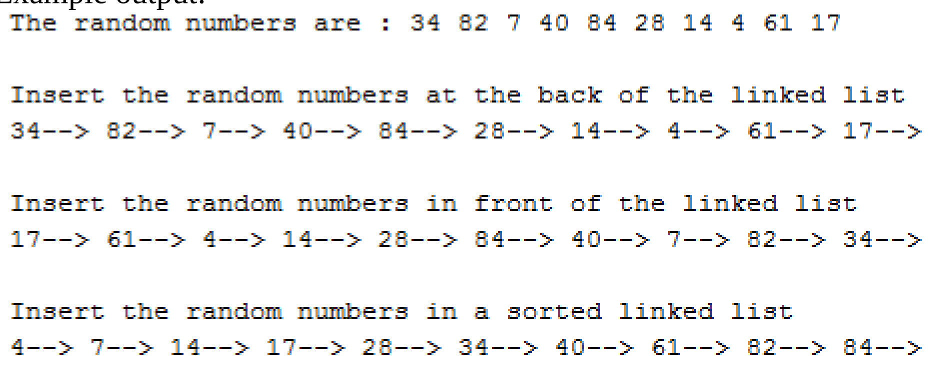
WIA1002 Data Structure
Lab 4: Linked List

1. Create the ListNode.java and LinkedList.java file (only the methods required based on the lecture notes). Then create a tester class that generate 10 random numbers within 0 – 100. After that, insert the numbers into the linked list as below:
    a. Insert the random number at the back of the linked list.
    b. Insert the random number in front of the linked list.
    c. Insert the random number in a sorted linked list. You need to create a new method addSortNode in the LinkedList.java.

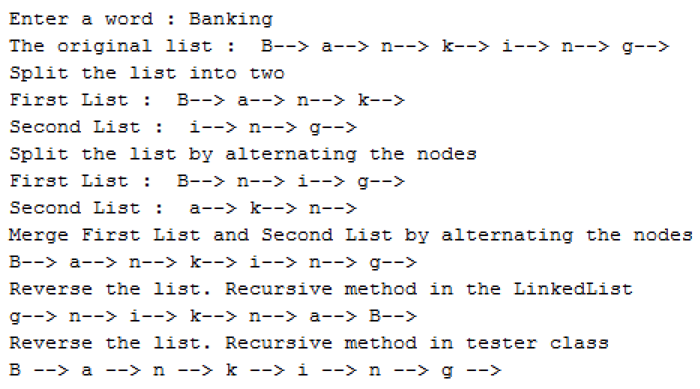
    Example output:



```
The random numbers are : 34 82 7 40 84 28 14 4 61 17

Insert the random numbers at the back of the linked list
34--> 82--> 7--> 40--> 84--> 28--> 14--> 4--> 61--> 17-->

Insert the random numbers in front of the linked list
17--> 61--> 4--> 14--> 28--> 84--> 40--> 7--> 82--> 34-->

Insert the random numbers in a sorted linked list
4--> 7--> 14--> 17--> 28--> 34--> 40--> 61--> 82--> 84-->
```

2. Write a program that inserts the characters of a sentence into a linked list. Then, perform the following:
    a. Create a new method **splitList** in the LinkedList.java. If the number of characters is odd, the extra character is inserted in the first list.
    b. Create a new method **alternateList** in the LinkedList.java. Split the list into two by alternating the elements from the original list.
    c. Create a new method **mergeList** in the LinkedList.java. Merge two lists into one by alternating the elements from both of the lists.
    d. Create a recursive method in the LinkedList.java to reverse the list.
    e. Create a static recursive method in the tester class to reverse the list.

    Example output:



```
Enter a word : Banking
The original list :  B--> a--> n--> k--> i--> n--> g-->
Split the list into two
First List :  B--> a--> n--> k-->
Second List :  i--> n--> g-->
Split the list by alternating the nodes
First List :  B--> n--> i--> g-->
Second List :  a--> k--> n-->
Merge First List and Second List by alternating the nodes
B--> a--> n--> k--> i--> n--> g-->
Reverse the list. Recursive method in the LinkedList
g--> n--> i--> k--> n--> a--> B-->
Reverse the list. Recursive method in tester class
B --> a --> n --> k --> i --> n --> g -->
```

3. Create a class Course that consists of the course code, course name, credit hours and the grade of the course (A, B, C, D and F). The point allocate to each grade is (4, 3, 2, 1 and 0). Then, insert the following information into a linked list. After that, calculate the grade point average for the student.

Course Code	**Course Name**	**Credit Hours**	**Grade**
WXX101	Programming	5	B
WXX201	Networking	4	C
WXX301	Operating System	3	A

Example output:

```
The list consist of
Course : WXX101 (Programming) - 5 credit hours. Grade : B -->
Course : WXX201 (Networking) - 4 credit hours. Grade : C -->
Course : WX3201 (Operating System) - 3 credit hours. Grade : A -->
Total point is 35
Total credit is 12
Grade point average is 2.92
```

4. Create a StarList Game. Each player will take turns to roll a dice. The first players that reach more than 20 stars in the lists win the game.

Example output:

```
Player 2 start first
Player 2: * --> * -->
Player 1: * --> * --> * --> * -->
Player 2: * --> * --> * --> * --> * --> * --> * -->
Player 1: * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 2: * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 1: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 2: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 1: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 2: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 1: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 2: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 1: * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * --> * -->
Player 1 wins the game
```

5. Iterator is used to cycle through a linked list to get the data of an element or remove an element. Create an inner class LinkedListIterator in LinkedList.java that implements the Iterator. Then, insert the following word ARS, CHE, LEI, MAN, LIV, TOT into the LinkedList. After that, demonstrate the use of the iterator methods to remove the word that consists of the A character. Create the listIterator method in the LinkedList.java to return the iterator

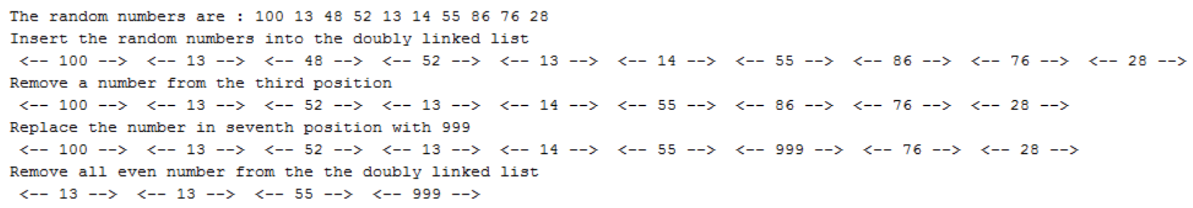
Example output:

```
The list consists of ARS --> AST --> CHE --> LEI --> MAN --> LIV --> TOT -->
Remove all the word that consists of the A character using iterator.
The updated list consists of CHE --> LEI --> LIV --> TOT -->
```

6. Create the DoubleListNode.java and DoubleLinkedList.java file (only the methods required based on the lecture notes). Then create a tester class that generate 10 random numbers within 0 – 100. After that, insert the numbers into the doubly linked list and perform the following actions:

    a. Remove a random number from third position.
    b. Replace the number in seventh position with 999.
    c. Remove all even number from the doubly linked list

    Example output:

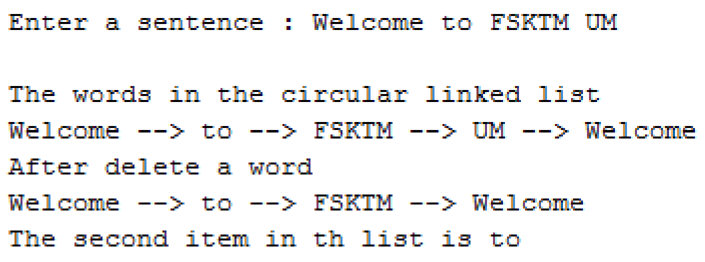


```
The random numbers are : 100 13 48 52 13 14 55 86 76 28
Insert the random numbers into the doubly linked list
 <-- 100 -->  <-- 13 -->  <-- 48 -->  <-- 52 -->  <-- 13 -->  <-- 14 -->  <-- 55 -->  <-- 86 -->  <-- 76 -->  <-- 28 -->
Remove a number from the third position
 <-- 100 -->  <-- 13 -->  <-- 52 -->  <-- 13 -->  <-- 14 -->  <-- 55 -->  <-- 86 -->  <-- 76 -->  <-- 28 -->
Replace the number in seventh position with 999
 <-- 100 -->  <-- 13 -->  <-- 52 -->  <-- 13 -->  <-- 14 -->  <-- 55 -->  <-- 999 -->  <-- 76 -->  <-- 28 -->
Remove all even number from the the doubly linked list
 <-- 13 -->  <-- 13 -->  <-- 55 -->  <-- 999 -->
```

7. Create the ListNode.java and CircularLinkedList.java file. The CircularLinkedList class must contains the following method:
    a. length()
    b. addCircularNode - insert at the back
    c. deleteCircularNode - delete from the back
    d. showCircularList
    e. getCicularItem – return the item by index

    Then, insert the words from a sentence to the CircularLinkedList. Then, delete a word from the list. After that, display the second item of the list.

    Example output:



```
Enter a sentence : Welcome to FSKTM UM

The words in the circular linked list
Welcome --> to --> FSKTM --> UM --> Welcome
After delete a word
Welcome --> to --> FSKTM --> Welcome
The second item in th list is to
```

8. Create a Music class that consists of the music title and the file name. Then, insert a few music object into the CircularLinkedList. After that, modify the code below to allow the users to play the music.

    Example code:

```
// code to play mp3 file for 10 seconds
JFXPanel panel = new JFXPanel();
String song = "just.mp3";
Media hit = new Media(Paths.get(song).toUri().toString());
MediaPlayer mediaPlayer = new MediaPlayer(hit);
mediaPlayer.play();
try {
   Thread.sleep(10000);
} catch (Exception e) {}
mediaPlayer.stop();
System.exit(0);
```

Example output:

```
My Music Play List
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 1
Play Music : one
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 2
Forward One Position - Play Music : two
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 2
Forward One Position - Play Music : three
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 3
Backward One Position - Play Music : two
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 2
Forward One Position - Play Music : three
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 2
Forward One Position - Play Music : four
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 2
Forward One Position - Play Music : one
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : 4
Stop Playing
Music : one --> Music : two --> Music : three --> Music : four --> Music : one
1 Play Music | 2 Forward | 3 Back | 4 Stop | -1 Exit : -1
Exit Music Player
```