Lab 7

||==||

Student Name: Faisal Majed Alghamdi

Student ID Number: 447001625

||==||

Student Name: Ammar Saad Alamri

Student Number: 447001588

||==||

Task 1 [.5 Mark] – Stack ADT

- **Balancing Symbols:** Write a method that uses a stack to check that every right brace, bracket, and parentheses must correspond to its matching left symbol – Examples:
  - Legal: "{[()]}", "([])" – the method returns true in this case.
  - Illegal: "[(])", ")[]", "([)]" or "((]" - the method returns false in this case.
- **Reversing a word or a sentence:** Write a method that uses a stack to display a given word in reverse order, e.g., “Data Structures” will be “serutcurtS ataD”.

```
public static boolean isBalance(String exp){  2 usages
    // ({})
    Stack<Character> stack = new Stack<>();
    for(int  i=0; i<exp.length(); i++){
        char ch = exp.charAt(i);
        if(ch=='(' || ch=='{' || ch=='['){
            stack.push(ch);
        }
        else if ((ch==')') || (ch==']') || ch=='}'){

            if(stack.isEmpty()){
                return false;
            }

            char top = stack.pop();

            if((ch==')' && top!='(')   || (ch==']' && top !='[') || (ch=='}' && top!='{'))
            {
                return false;
            }
        } }
    return stack.isEmpty();
}
```

Output

```
||=========================================||
|| Student Name: Faisal Majed Alghamdi     ||
|| Student Number: 447001625               ||
||=========================================||
||=========================================||
|| Student Name: AmmarSaadALamri           ||
|| Student Number: 447001588               ||
||=========================================||



()(){}[[()]] is : true
()(){}{[]}] is : false
```

- **Reversing a word or a sentence:** Write a method that uses a stack to display a given word in reverse order, e.g., “Data Structures” will be “serutcurtS ataD”.

```java
public static String revise(String word){  1 usage

    Stack<Character> stack = new Stack<>();

    for(int i = 0; i<word.length(); i++){
        stack.push(word.charAt(i));
    }

    String result = "";

    while(!stack.isEmpty()){

        char temp = stack.pop();
        result = result+ temp;
    }

    return result;

}
```

Output

```
||=========================================||
|| Student Name: Faisal Majed Alghamdi     ||
|| Student Number: 447001625               ||
||=========================================||
||=========================================||
|| Student Name: AmmarSaadALamri           ||
|| Student Number: 447001588               ||
||=========================================||



erutcurtS ataD
```

Task 2 [.5 Mark] - Queue ADT

The following Java files are provided:

- Queue.java – Queue ADT interface.
- ArrayQueue.java – Array-based Queue implementation.
- LinkedListQueue.java – Linked-list-based Queue implementation.

Sample Output

```
=== ArrayQueue<Integer> ===
Queue: [10, 20, 30]
First: 10
Dequeue: 10
Queue after enqueue: [20, 30, 40, 50]
Size: 4
Is empty? false

=== LinkedListQueue<String> ===
Queue: [Sara, Nora, Lama]
First: Sara
Dequeue: Sara
Queue after dequeue: [Nora, Lama]
Size: 2
Is empty? false
```

Instructions:

1. Read and understand the provided Java classes.
2. Create a test program named testProgram.java.
3. In your test program:
   - Create an ArrayQueue of integers.
   - Create a LinkedListQueue of strings.
   - Test the following operations on both queues: enqueue(), dequeue(), first(), size(), and isEmpty().
   - Display the queue elements using toString().

Part2

```java
public static void main(String[] args) {


    System.out.println("||========================================||");
    System.out.printf("|| %-40s ||%n","Student Name: Faisal Majed Alghamdi");
    System.out.printf("|| %-40s ||%n","Student Number: 447001625");
    System.out.println("||========================================||");
    System.out.println("||========================================||");
    System.out.printf("|| %-40s ||%n","Student Name: AmmarSaadALamri");
    System.out.printf("|| %-40s ||%n","Student Number: 447001588");
    System.out.println("||========================================||\n\n\n");

    Queue<Integer> intQueue = new ArrayQueue<>();

    System.out.println("=========ArryQueue<Integer>==========");

    intQueue.enqueue( e: 10);
    intQueue.enqueue( e: 20);
    intQueue.enqueue( e: 30);

    System.out.println("Queue: "+intQueue.toString());
    System.out.println("First: " + intQueue.first());
    System.out.println("Dequeu: " + intQueue.dequeue());
    intQueue.enqueue( e: 40);
    intQueue.enqueue( e: 50);
    System.out.print("Queue after enqueue : " + intQueue.toString()+ '\n');
    System.out.println("size : " + intQueue.size());
    System.out.println("isEmpty" + intQueue.isEmpty());
```

```java
System.out.println("\n\n===============LinkedList Queue==================");


Queue<String> strQueue= new LinkedListQueue<>();

strQueue.enqueue( e: "Sara");
strQueue.enqueue( e: "Nora");
strQueue.enqueue( e: "Lama");
System.out.println("Queue = " + strQueue.toString());

System.out.println("First : " + strQueue.first());
System.out.println("Dequeue : " + strQueue.dequeue());
System.out.println("Queue after dequeue = " + strQueue.toString());

System.out.println("Size : " + strQueue.size());
System.out.println("is Empty : " + strQueue.isEmpty());
```

Output

```
==========ArryQueue<Integer>==========
Queue: [10, 20, 30]
First: 10
Dequeu: 10
Queue after enqueue : [20, 30, 40, 50]
size : 4
isEmptyfalse


===============LinkedList Queue==================
Queue = [Sara, Nora, Lama]
First : Sara
Dequeue : Sara
Queue after dequeue = [Nora, Lama]
Size : 2
is Empty : false
```