```
import java.util.LinkedList;
import java.util.Queue;

class process {
    int arrTime;
    int exeTime;
    process(int arr, int exe) {
        arrTime = arr; exeTime = exe;
    }
}
public class RoundRobinScheduling {
    public static float Solution(int[] Atime, int[] Etime, int q) {
        if (Atime == null || Etime == null || Atime.length != Etime.length)
            return 0;
        int length = Atime.length;
        Queue<process> queue = new LinkedList<process>();
        int curTime = 0, waitTime = 0;
        int index = 0;
        while (!queue.isEmpty() || index < length) {
            if (!queue.isEmpty()) {
                process cur = queue.poll();
                waitTime += curTime - cur.arrTime;
                curTime += Math.min(cur.exeTime, q);
                for (; index < length && Atime[index] <= curTime; index++)
                    queue.offer(new process(Atime[index], Etime[index]));
                if (cur.exeTime > q)
                    queue.offer(new process(curTime, cur.exeTime - q));
            }
            else {
                queue.offer(new process(Atime[index], Etime[index]));
                curTime = Atime[index++];
            }
        }
        return (float) waitTime / length;
    }
}
```