Ejercicio 1. Recursividad

Dado la siguiente implementación funcional:

```
public static Long mistral(int n, int k) {
	return LongStream.rangeClosed(1l, k)
			.filter(e -> e%2==0)
			.map(e-> (n-e +1) / e)
			.sum();
}
```

SE PIDE (ADDA):
1) Implementar la versión iterativa en C.
2) Implementar la versión recursiva final en C.

SE PIDE (EDA):
1) Implementar la versión iterativa en Java.
2) Implementar la versión recursiva final en Java.

Solución en C

```
long mistralFinalAux(int n,  int k, int e, long ac) {
	if (e<=k) {
		if (e%2==0) {
		  ac = mistralFinalAux(n,  k, e+1,  ac +  ((n-e +1) / e));
		} else {
			ac = mistralFinalAux(n,  k, e+1,  ac);
		}
	}
	return ac;
}

long mistralFinal(int n, int k) {
    return mistralFinalAux(n, k, 1, 0l);
}

long mistralIter(int n, int k) {
	long acum = 0l;
	int e = 1;
	while (e<=k) {
		if (e%2==0) {
			long t = (long)(n-e +1) / e;
			acum = acum + t;
		}
		e = e + 1;
	}
	return acum;
}
```

Solución en Java

```
public static Long mistralFinal(int n, int k) {
    return mistralFinal(n, k, 1, 0l);
}

private static Long mistralFinal(int n,  int k, int e, Long ac) {
	if (e<=k) {
		if (e%2==0) {
		  ac = mistralFinal(n,  k, e+1,  ac +  ((n-e +1) / e));
		} else {
			ac = mistralFinal(n,  k, e+1,  ac);
		}
	}
	return ac;
}


public static Long mistralIter(int n, int k) {
	Long acum = 0l;
	Integer e = 1;
	while (e<=k) {
		if (e%2==0) {
			Long t = (long)(n-e +1) / e;
			acum = acum + t;
		}
		e = e + 1;
	}
	return acum;
}
```