**Mark and Toys**

**Solution**

**static** **int** maximumToys(**int**[] prices, **int** k) {

**int** count = 0, sum = 0;

Arrays.*sort*(prices);

**for** (**int** i : prices) {

**if** (sum + i <= k) {

sum += i;

count++;

}

}

**return** count;

}

**1. The complexity**

**Time Complexity:** O(n log(n)) // to run quicksort on the input array

**Space Complexity:** O(n) // dynamic allocation of array to store the input

**The forEach loop Complexity is :** O(n)

**Total Complexity =** O(n log(n)) + O(n)

**2. Data structures/algorithms**

Arrays.sort() internally uses a QuickSort algorithm

**3. Design patterns**

Strategy Design Pattern

**4. Is your provided solution efficient for any given input data range?** Yes

**Test Cases 1**

7 50

1 12 5 111 200 1000 10

Result: 4

**Test Cases 2**

4 7

1 2 3 4

Result: 3

**Test Cases 3**

5 15

3 7 2 9 4

Result: 3

**Test Cases 4**

15 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Result: 13