Aplicaţi criteriul de testare white-box pentru stabilirea cazurilor de testare.

I. Metoda `isPrime` verifică dacă un număr natural n dat este prim.

	`/**`
	` * @param n natural number`
	` * @return true, false`
	` * @exception InvalidValueException if n < 0`
	` */`

	`private boolean isPrime(int n) throws InvalidValueException{`
1	`    boolean result = true;`
2	`    if (n<0)`
3	`        throw new InvalidValueException("negative value was provided!");`
4	`    else if ((n == 0)||(n == 1))`
5	`        result = false;`
	`    else{`
6	`        int d = 2;`
7	`        while (d<=Math.ceil(n/2)){`
8	`            if (n%d == 0)`
9	`                result = false;`
10	`            d++;`
	`        }`
	`    }`
11	`    return result;`
12	`}`

II. Metoda `computeMaxCounter` determină frecvenţa de apariţie a valorii maxime dintr-o listă de elemente date.

	`/**`
	` * @param list - lista de valori intregi`
	` * @return frecventa de aparitie a valorii maxime`
	` */`

	`public int computeMaxCounter(List<Integer> list){`
1	`   int countMax, index, posMax;`
	`   index=0;countMax=0;posMax=0;`
2	`   while (index<list.size()){`
3	`      if (list.get(index) > list.get(posMax)){`
4	`         posMax=index;`
	`         countMax=1;`
	`      }`
	`      else`
5	`         if (list.get(posMax)== list.get(index))`
6	`            countMax++;`
7	`      index++;`
	`   }`
8	`   if(list.size()==0)`
9	`      return -1;`
10	`   else return countMax;`
11	`}`