1===

chamada	condição de parada
fat(n-1)	n == 1
fib(n-1); fib(n-2)	n == 0 || n == 1

2===

Porque há duas instruções **System.out.println(i)**: uma antes da chamada recursiva e outra depois.

3===

```
int multiplicacao (int a, int b) {
     int resp = 0;
     if (b > 0) {
          resp = a + multiplicacao(a, b - 1);
     }
     return resp;
}

void main (...) {
     multiplicacao(5, 13);
}
```

4===

```
int maior (int vet[], int n) {
     return maior (vet, n, 0);
}

int maior (int vet[], int n, int i) {
     int resp;
     if (i == n – 1) {
          resp = vet[n - 1];
     } else {
          resp = maior(vet, n, i + 1);

          if (resp < vet[i]) {
               resp = vet[i];
          }
     }
     return resp;
}
```

5===

```
boolean isPalindromo(String s) {
     return isPalindromo(s, 0);
}

boolean isPalindromo(String s, int i) {
     boolean resp;
     if (i >= s.length() / 2) {
          resp = true;
     } else if (s.charAt(i) != s.charAt(s.length() - 1 -  i)) {
          resp = false;
     } else {
          resp = isPalindromo(s, i + 1);
     }
     return resp;
}
```

6===

```
boolean isVowel(char c) {
     return (c == ‘a’ || c == ‘e’ || c == ‘i’ ||
             c == ‘o’ || c == ‘u’);
}

int countVowels(String s, int i) {
     int count = 0;

     if (i < s.length()) {
          if (isVowel(s.charAt(i))) {
               count++;
          }
     count += countVowels(s, i+1);
     }
     return count;
}

void main(...) {
     countVowels(s, 0);
}
```

7===

```
boolean isLower(char c) {
     return (‘a’ <= c && c <= ‘z’);
}

boolean isVowel(char c) {
     return (c == ‘a’ || c == ‘e’ || c == ‘i’ ||
             c == ‘o’ || c == ‘u’);
}

int countChars(String s, int i) {
     int count = 0;

     if (i < s.length()) {
          if (isLower(s.charAt(i)) && !isVowel(s.charAt(i)) {
               count++;
          }
     count += countChars(s, i+1);
     }
     return count;
}

void main(...) {
     countChars(s, 0);
}
```

8===

```
void insertionSort(int[] a, int n) {
    if (n <= 1) {
        return;
    }

    insertionSort(a, n-1);
    int m = a[n-1];
    int i = n - 2;

    while (i >= 0 && a[i] > m) {
        a[i+1] = a[i];
        i--;
    }

    a[i+1]= m;
}

void main(...) {
    int[] array = {...};
    insertionSort(array, array.length);
}
```

9===

```
int T(int n) {
    if (n == 0) {
        return 1;
    } else if (n == 1) {
        return 2;
    } else {
        return T(n-1) * T(n-2) - T(n-1);
    }
}
```

10===

```
int T(int n) {
    if (n == 0) {
        return 1;
    } else {
        return (int) Math.pow(T(n-1), 2);
    }
}
```