```c
#include <stdio.h>
#include <assert.h>

1       unsigned int maxevenrec(unsigned int arr[],
2                               int index, int size) {
3         int evenValue_1 = 0;
4         int evenValue_2 = 0;
5
6         if(size == 0) {
7               return 0;
8         }
9
10
11        if(arr[index] %2 == 0) {
12              evenValue_1 = arr[index];
13        }
14
15        evenValue_2 = maxevenrec(arr, index+1, size-1);
16
17        if (evenValue_1 > evenValue_2) {
18              return evenValue_1;
19        }
20        return evenValue_2;
21      }
22
23      int main(){
24        unsigned int arr[15] = {1, 4};
25
26        assert(maxevenrec(arr, 0, 2) == 4);
27      }
```

1. Durchlauf

Funktionsaufruf
Zeile 1: `arr = {1, 4}, index = 0, size = 2`

Zeilen 3 & 4: Die lokalen Variablen `evenValue_1` und `evenValue_2` werden bei jedem Aufruf auf `0` gesetzt

Zeile 6: `false`, wird nicht aufgerufen

Zeile 11: `arr[0]` ist `1`, daher ist die Abfrage false und `evenValue_1` bleibt `0`

Zeile 15: Rekursiver Aufruf
++++++++++++++++++++++++++++++++++
2. Durchlauf

Funktionsaufruf
Zeile 1: `arr = {1, 4}, index = 1, size = 1`

Zeile 6: `false`, wird nicht aufgerufen

Zeile 11: `arr[1]` ist `4`, daher ist die Abfrage true und `evenValue_1` wird auf `4` gesetzt

Zeile 15: Rekursiver Aufruf
++++++++++++++++++++++++++++++++++

3. Durchlauf

Funktionsaufruf
Zeile 1: `arr = {1, 4}, index = 2, size = 0`

Zeile 6: `true`, gibt 0 zurück und beendet den Funktionsaufruf
++++++++++++++++++++++++++++++++++

(zurück zum 2. Durchlauf)
Zeile 15: `evenValue_2` ist `0`

Zeile 17: `evenValue_1 (4)` ist größer als `evenValue_2 (0)`, daher wird `4` zurückgegeben
++++++++++++++++++++++++++++++++++

(zurück zum 1. Durchlauf)
Zeile 15: `evenValue_2` ist `4`

Zeile 17: `evenValue_1 (0)` ist kleiner als `evenValue_2 (4)`, daher wird `4` zurückgegeben
++++++++++++++++++++++++++++++++++

Der rekursive Aufruf wurde rückabgewickelt und die Funktion gibt `4` zurück