```cpp
 1: // $Id: listfree.cpp,v 1.20 2015-03-26 22:10:36-07 - - $
 2:
 3: // Show how to break a cycle in a simple circular list.
 4:
 5: #include <algorithm>
 6: #include <iostream>
 7: #include <memory>
 8: using namespace std;
 9:
10: struct node;
11:
12: using node_ptr = shared_ptr<node>;
13:
14: struct node {
15:    int value;
16:    node_ptr link;
17:    node (int value, node_ptr link): value(value), link(link) {}
18: };
19:
20: int main (int argc, char** argv) {
21:    cout << "Command:";
22:    for_each (&argv[0], &argv[argc], [](char* arg){cout << " " << arg;});
23:    cout << endl;
24:    bool break_cycle = argc > 1 and string (argv[1]) == "-f";
25:    node_ptr list = make_shared<node> (1,
26:                 make_shared<node> (2,
27:                 make_shared<node> (3, nullptr)));
28:    list->link->link->link = list;
29:    cout << "list = " << list << endl;
30:    for (auto curr = list;;) {
31:       cout << curr << " -> {" << curr->value << ", " << curr->link
32:            << "} (use_count " << curr.use_count() << ")" << endl;
33:       curr = curr->link;
34:       if (curr == list) break;
35:    }
36:    if (break_cycle) list->link = nullptr;
37:    return 0;
38: }
39:
40: //TEST// valgrind listfree -0 >listfree.out-0 2>&1
41: //TEST// valgrind listfree -f >listfree.out-f 2>&1
42: //TEST// mkpspdf listfree.ps listfree.cpp* listfree.out-*
43:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting listfree.cpp
2: listfree.cpp:
3:      $Id: listfree.cpp,v 1.20 2015-03-26 22:10:36-07 - - $
4: g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++11 listfree.cpp -o listfree
-lglut -lGLU -lGL -lX11 -lrt -lm
5: rm -f listfree.o
6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished listfree.cpp
```

```
 1: ==13663== Memcheck, a memory error detector
 2: ==13663== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
 3: ==13663== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright i
nfo
 4: ==13663== Command: listfree -0
 5: ==13663==
 6: Command: listfree -0
 7: list = 0x4e7d1e8
 8: 0x4e7d1e8 -> {1, 0x4e7d178} (use_count 3)
 9: 0x4e7d178 -> {2, 0x4e7d108} (use_count 2)
10: 0x4e7d108 -> {3, 0x4e7d1e8} (use_count 2)
11: ==13663==
12: ==13663== HEAP SUMMARY:
13: ==13663==     in use at exit: 144 bytes in 3 blocks
14: ==13663==   total heap usage: 5 allocs, 2 frees, 180 bytes allocated
15: ==13663==
16: ==13663== LEAK SUMMARY:
17: ==13663==    definitely lost: 48 bytes in 1 blocks
18: ==13663==    indirectly lost: 96 bytes in 2 blocks
19: ==13663==      possibly lost: 0 bytes in 0 blocks
20: ==13663==    still reachable: 0 bytes in 0 blocks
21: ==13663==         suppressed: 0 bytes in 0 blocks
22: ==13663== Rerun with --leak-check=full to see details of leaked memory
23: ==13663==
24: ==13663== For counts of detected and suppressed errors, rerun with: -v
25: ==13663== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```

```
 1: ==13664== Memcheck, a memory error detector
 2: ==13664== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
 3: ==13664== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright i
nfo
 4: ==13664== Command: listfree -f
 5: ==13664==
 6: Command: listfree -f
 7: list = 0x4e7d1e8
 8: 0x4e7d1e8 -> {1, 0x4e7d178} (use_count 3)
 9: 0x4e7d178 -> {2, 0x4e7d108} (use_count 2)
10: 0x4e7d108 -> {3, 0x4e7d1e8} (use_count 2)
11: ==13664==
12: ==13664== HEAP SUMMARY:
13: ==13664==     in use at exit: 0 bytes in 0 blocks
14: ==13664==   total heap usage: 5 allocs, 5 frees, 180 bytes allocated
15: ==13664==
16: ==13664== All heap blocks were freed -- no leaks are possible
17: ==13664==
18: ==13664== For counts of detected and suppressed errors, rerun with: -v
19: ==13664== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```