```
 1: // $Id: listfree.cpp,v 1.24 2019-01-10 13:58:49-08 - - $
 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 argv[1] == "-f"s;
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: checksource listfree.cpp
    3: ident listfree.cpp
    4: listfree.cpp:
    5:      $Id: listfree.cpp,v 1.24 2019-01-10 13:58:49-08 - - $
    6: cpplint.py.perl listfree.cpp
    7: Done processing listfree.cpp
    8: g++ -g -O0 -Wall -Wextra -Wpedantic -Wshadow -fdiagnostics-color=never -
std=gnu++17 -Wold-style-cast listfree.cpp -o listfree -lm
    9: rm -f listfree.o
   10: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished listfree.cpp
```

```
    1: ==31767== Memcheck, a memory error detector
    2: ==31767== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
    3: ==31767== Using Valgrind-3.14.0.GIT and LibVEX; rerun with -h for copyri
ght info
    4: ==31767== Command: listfree -0
    5: ==31767==
    6: Command: listfree -0
    7: list = 0x5a23190
    8: 0x5a23190 -> {1, 0x5a23120} (use_count 3)
    9: 0x5a23120 -> {2, 0x5a230b0} (use_count 2)
   10: 0x5a230b0 -> {3, 0x5a23190} (use_count 2)
   11: ==31767==
   12: ==31767== HEAP SUMMARY:
   13: ==31767==     in use at exit: 120 bytes in 3 blocks
   14: ==31767==   total heap usage: 4 allocs, 1 frees, 147 bytes allocated
   15: ==31767==
   16: ==31767== LEAK SUMMARY:
   17: ==31767==    definitely lost: 40 bytes in 1 blocks
   18: ==31767==    indirectly lost: 80 bytes in 2 blocks
   19: ==31767==      possibly lost: 0 bytes in 0 blocks
   20: ==31767==    still reachable: 0 bytes in 0 blocks
   21: ==31767==         suppressed: 0 bytes in 0 blocks
   22: ==31767== Rerun with --leak-check=full to see details of leaked memory
   23: ==31767==
   24: ==31767== For counts of detected and suppressed errors, rerun with: -v
   25: ==31767== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
    1: ==31771== Memcheck, a memory error detector
    2: ==31771== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
    3: ==31771== Using Valgrind-3.14.0.GIT and LibVEX; rerun with -h for copyri
ght info
    4: ==31771== Command: listfree -f
    5: ==31771==
    6: Command: listfree -f
    7: list = 0x5a23190
    8: 0x5a23190 -> {1, 0x5a23120} (use_count 3)
    9: 0x5a23120 -> {2, 0x5a230b0} (use_count 2)
   10: 0x5a230b0 -> {3, 0x5a23190} (use_count 2)
   11: ==31771==
   12: ==31771== HEAP SUMMARY:
   13: ==31771==     in use at exit: 0 bytes in 0 blocks
   14: ==31771==   total heap usage: 4 allocs, 4 frees, 147 bytes allocated
   15: ==31771==
   16: ==31771== All heap blocks were freed -- no leaks are possible
   17: ==31771==
   18: ==31771== For counts of detected and suppressed errors, rerun with: -v
   19: ==31771== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```