```
 1: // $Id: listfree.cpp,v 1.23 2016-04-13 15:56:27-07 - - $
 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 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: listfree.cpp:
3:      $Id: listfree.cpp,v 1.23 2016-04-13 15:56:27-07 - - $
4: g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++14 listfree.cpp
5:          -o listfree -lglut -lGLU -lGL -lX11 -lrt -lm
6: rm -f listfree.o
7: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished listfree.cpp
```

```
    1: ==21140== Memcheck, a memory error detector
    2: ==21140== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
    3: ==21140== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
    4: ==21140== Command: listfree -0
    5: ==21140==
    6: Command: listfree -0
    7: list = 0x9c9a1e0
    8: 0x9c9a1e0 -> {1, 0x9c9a170} (use_count 3)
    9: 0x9c9a170 -> {2, 0x9c9a100} (use_count 2)
   10: 0x9c9a100 -> {3, 0x9c9a1e0} (use_count 2)
   11: ==21140==
   12: ==21140== HEAP SUMMARY:
   13: ==21140==     in use at exit: 120 bytes in 3 blocks
   14: ==21140==   total heap usage: 5 allocs, 2 frees, 163 bytes allocated
   15: ==21140==
   16: ==21140== LEAK SUMMARY:
   17: ==21140==    definitely lost: 40 bytes in 1 blocks
   18: ==21140==    indirectly lost: 80 bytes in 2 blocks
   19: ==21140==      possibly lost: 0 bytes in 0 blocks
   20: ==21140==    still reachable: 0 bytes in 0 blocks
   21: ==21140==         suppressed: 0 bytes in 0 blocks
   22: ==21140== Rerun with --leak-check=full to see details of leaked memory
   23: ==21140==
   24: ==21140== For counts of detected and suppressed errors, rerun with: -v
   25: ==21140== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```

```
    1: ==21164== Memcheck, a memory error detector
    2: ==21164== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
    3: ==21164== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
    4: ==21164== Command: listfree -f
    5: ==21164==
    6: Command: listfree -f
    7: list = 0x9c9a1e0
    8: 0x9c9a1e0 -> {1, 0x9c9a170} (use_count 3)
    9: 0x9c9a170 -> {2, 0x9c9a100} (use_count 2)
   10: 0x9c9a100 -> {3, 0x9c9a1e0} (use_count 2)
   11: ==21164==
   12: ==21164== HEAP SUMMARY:
   13: ==21164==     in use at exit: 0 bytes in 0 blocks
   14: ==21164==   total heap usage: 5 allocs, 5 frees, 163 bytes allocated
   15: ==21164==
   16: ==21164== All heap blocks were freed -- no leaks are possible
   17: ==21164==
   18: ==21164== For counts of detected and suppressed errors, rerun with: -v
   19: ==21164== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```