```
 1: // $Id: uninitnew.cpp,v 1.2 2014-04-22 18:55:00-07 - - $o
 2:
 3: //
 4: // Illustrates the uninitialized behavior of areas returned by
 5: // new when there is no default ctor, as for primitives and pointers.
 6: // Allocate an array, print non-zero entries, then free the array.
 7: // check for memory leak.
 8: //
 9:
10: #include <iostream>
11: #include <string>
12: #include <vector>
13:
14: using namespace std;
15:
16: void f(int n) {
17:    cout << n << ":";
18:    int *p = new int[n];
19:    for (int i = 0; i < n; ++i) {
20:       if (p[i]!=0) cout << " " << dec << i << "=" << hex << p[i] << ";";
21:       p[i] = 0xDEADBEEF;
22:    }
23:    cout << endl;
24:    delete[] p;
25: }
26:
27: void g() {
28:    vector<int*> vi(5);
29:    cout << "g:";
30:    for (size_t i = 0; i < vi.size(); ++i) cout << " " << vi[i];
31:    cout << endl;
32: }
33:
34: int main() {
35:    f(10);
36:    f(5);
37:    f(6);
38:    f(100);
39:    f(8);
40:    g();
41:    return 0;
42: }
43:
44: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
45: //TEST// grind uninitnew >uninitnew.out 2>&1
46: //TEST// mkpspdf uninitnew.ps uninitnew.cpp* uninitnew.out
47:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting uninitnew.cpp
2: uninitnew.cpp: $Id: uninitnew.cpp,v 1.2 2014-04-22 18:55:00-07 - - $
3: g++ -g -O0 -Wall -Wextra -std=gnu++11 uninitnew.cpp -o uninitnew -lm
4: rm -f uninitnew.o
5: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished uninitnew.cpp
```

```
==31179== Memcheck, a memory error detector
==31179== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al
.
==31179== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright i
nfo
==31179== Command: uninitnew
==31179==
==31179== Conditional jump or move depends on uninitialised value(s)
==31179==    at 0x400B99: f(int) (uninitnew.cpp:20)
==31179==    by 0x400D58: main (uninitnew.cpp:35)
==31179==
10:
==31179== Conditional jump or move depends on uninitialised value(s)
==31179==    at 0x400B99: f(int) (uninitnew.cpp:20)
==31179==    by 0x400D62: main (uninitnew.cpp:36)
==31179==
5:
==31179== Conditional jump or move depends on uninitialised value(s)
==31179==    at 0x400B99: f(int) (uninitnew.cpp:20)
==31179==    by 0x400D6C: main (uninitnew.cpp:37)
==31179==
6:
==31179== Conditional jump or move depends on uninitialised value(s)
==31179==    at 0x400B99: f(int) (uninitnew.cpp:20)
==31179==    by 0x400D76: main (uninitnew.cpp:38)
==31179==
100:
==31179== Conditional jump or move depends on uninitialised value(s)
==31179==    at 0x400B99: f(int) (uninitnew.cpp:20)
==31179==    by 0x400D80: main (uninitnew.cpp:39)
==31179==
8:
g: 0 0 0 0 0
==31179==
==31179== HEAP SUMMARY:
==31179==     in use at exit: 0 bytes in 0 blocks
==31179==   total heap usage: 6 allocs, 6 frees, 556 bytes allocated
==31179==
==31179== All heap blocks were freed -- no leaks are possible
==31179==
==31179== For counts of detected and suppressed errors, rerun with: -v
==31179== Use --track-origins=yes to see where uninitialised values come
 from
==31179== ERROR SUMMARY: 129 errors from 5 contexts (suppressed: 6 from
6)
```