Using valgrind to find memory leaks

The valgrind utility can be used to detect memory leaks in your C programs. To run it type in "valgrind" followed by the parameter setting "**`--leak-check=full`**", followed by the usual command you would use to run your program. For example, to run a program called "myProgram", passing it a value of 42 as a runtime parameter, type the following:

```
valgrind  --leak-check=full  myProgram  42
```

Your program will run more slowly than usual, but at the end valgrind will output a report showing any memory leaks. A simple example is shown below.

No Memory Leak

Here is my program, called freeTest, which does not contain a memory leak:

```
#include <stdlib.h>

int main ()
{
   int   *p = (int *)malloc(20*sizeof(int));
   free(p);
   return 0;
}
```

This is the valgrind command to run the program:

```
valgrind --leak-check=full freeTest
```

and here is the resulting output, with the important bits highlighted:

```
==5141== Memcheck, a memory error detector
==5141== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==5141== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==5141== Command: freeTest
==5141==
==5141==
==5141== HEAP SUMMARY:
==5141==     in use at exit: 0 bytes in 0 blocks
==5141==   total heap usage: 1 allocs, 1 frees, 80 bytes allocated
==5141==
==5141== All heap blocks were freed -- no leaks are possible
==5141==
==5141== For counts of detected and suppressed errors, rerun with: -v
==5141== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```

Memory Leak

Here is the same program, but a memory leak introduced by not `free`-ing the pointer p:

```
#include <stdlib.h>

int main ()
{
   int   *p = (int *)malloc(20*sizeof(int));
   /* oops! forgot to free p */
   return 0;
}
```

The valgrind command is the same as before:

```
valgrind --leak-check=full freeTest
```

and here is the resulting output, again with the important bits highlighted:

```
==4802== Memcheck, a memory error detector
==4802== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==4802== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==4802== Command: freeTest
==4802==
==4802==
==4802== HEAP SUMMARY:
==4802==     in use at exit: 80 bytes in 1 blocks
==4802==   total heap usage: 1 allocs, 0 frees, 80 bytes allocated
==4802==
==4802== 80 bytes in 1 blocks are definitely lost in loss record 1 of 1
==4802==    at 0x4A069EE: malloc (vg_replace_malloc.c:270)
==4802==    by 0x400505: main (in /secamfs/userspace/phd/jtc202/c/ecm2424/freeTest)
==4802==
==4802== LEAK SUMMARY:
==4802==    definitely lost: 80 bytes in 1 blocks
==4802==    indirectly lost: 0 bytes in 0 blocks
==4802==      possibly lost: 0 bytes in 0 blocks
==4802==    still reachable: 0 bytes in 0 blocks
==4802==         suppressed: 0 bytes in 0 blocks
==4802==
==4802== For counts of detected and suppressed errors, rerun with: -v
==4802== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 6 from 6)
```

Note that it not only tells you how much memory has leaked, but also the place in the program where the leaked memory was allocated.