

## CoCoALib - Bug #790

### RingDistrMPolyInIFpPPImpl::mySummandPool frees ZERO PTR many times

28 Oct 2015 11:12 - John Abbott

<b>Status:</b>	Closed	<b>Start date:</b>	28 Oct 2015
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	John Abbott	<b>% Done:</b>	100%
<b>Category:</b>	Tidying	<b>Estimated time:</b>	1.51 hour
<b>Target version:</b>	CoCoALib-0.99550 spring 2017	<b>Spent time:</b>	1.25 hour
<b>Description</b>			
I have run test-SparsePolyRing1 with MemPool verbose active, and there were lots of warnings about freeing ZERO PTR. This is probably not right -- investigate!			

#### History

##### #1 - 28 Oct 2015 11:15 - John Abbott

I compiled CoCoALib with the MemPool debugging options active.  
In test-SparsePolyRing1.C I inserted the following line immediately before creating the GlobalManager

```
MemPoolDebug::ourInitialVerbosityLevel = 2;
```

The output is about 2800 lines long of which about 2000 lines are warnings about freeing ZERO PTR.

##### #2 - 28 Oct 2015 15:25 - John Abbott

- Status changed from New to Resolved
- Assignee set to John Abbott
- % Done changed from 0 to 70

The problem was a missing pair of curly brackets around a "then" clause comprising two commands (location DistrMPolyInIFpPP.H:87). Not sure if this is code I have recently "hacked", or whether the bug has been around for a while -- I cannot access CVS at the moment.

##### #3 - 29 Oct 2015 11:56 - John Abbott

Now I am slightly undecided which implementation I prefer:  
Impl (A):

```
~NewSummandPtr()  
{ if (myPtr == 0/*nullptr*/) return;  
  myPtr->~summand();  
  myMemMgr.free(myPtr);  
}
```

Impl (B):

```
~NewSummandPtr()  
{ if (myPtr != 0/*nullptr*/)
```

```
{ myPtr->~summand();  
  myMemMgr.free(myPtr);  
}
```

Originally I had written impl **(B)** (but had forgotten the curly brackets around the "then"-part). Now I have rewritten it as impl **(A)**, which I think is slightly easier to read...

Opinions?

#### #4 - 29 Oct 2015 14:52 - Anna Maria Bigatti

John Abbott wrote:

Now I am slightly undecided which implementation I prefer:

Impl **(A)**:

[...]

Impl **(B)**:

[...]

Originally I had written impl **(B)** (but had forgotten the curly brackets around the "then"-part). Now I have rewritten it as impl **(A)**, which I think is slightly easier to read...

Opinions?

I prefer A.

I think it is easier to read. I wonder if it makes a difference in execution time.

#### #5 - 07 Nov 2016 13:31 - John Abbott

- Status changed from Resolved to Closed

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99550 spring 2017

- % Done changed from 70 to 100

This was resolved a year ago. I have just performed the test suggested in comment 1, and there are no warnings (the output is about 770 lines, and all appears to be fine). So I regard this as fully solved ==> closing.

#### #6 - 28 Apr 2017 09:32 - Anna Maria Bigatti

- Estimated time set to 1.51 h