CoCoALib - Bug #1256

RingID: different values in test-output on different platforms

15 Mar 2019 17:56 - John Abbott

Status:	Closed	**Start date:**	15 Mar 2019
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Portability	**Estimated time:**	2.01 hours
Target version:	CoCoALib-0.99650 November 2019	**Spent time:**	0.90 hour

Description

In test-output Anna and I get different outputs (due to different RingIDs) because we use different platforms.

Investigate, and decide how to fix.

Related issues:

Related to CoCoA-5 - Support #1240: John's visit Feb 2019	**Closed**	**08 Feb 2019**
Related to CoCoALib - Design #1225: Move to C++14 (skipping C++11)	**In Progress**	**06 Sep 2018**
Related to CoCoALib - Feature #1249: principal ideal has a Gbasis	**Closed**	**01 Mar 2019**

History

#1 - 15 Mar 2019 17:59 - John Abbott

Anna uses clang on MacOS, while I use g++ on Linux.
We get different outputs for test-output: its seems that the RingIDs increase faster on Linux (as though more "temporary rings" are created internally somewhere).

A concrete example is:

```
NewQuotientRing(R,ideal(RingElem(R, "a^2-2")))
```

Anna sees the RingID increase by 1; John sees it increase by 2.

#2 - 15 Mar 2019 17:59 - John Abbott

- Related to Support #1240: John's visit Feb 2019 added

#3 - 15 Mar 2019 18:00 - John Abbott

- Related to Design #1225: Move to C++14 (skipping C++11) added

#4 - 27 Mar 2019 15:11 - John Abbott

```
ring P = NewPolyRing(RingQQ(), symbols("a"));
ring PmodI = NewQuotientRing(P, ideal(RingElem(P,"a^2-2")));
cout << "P = " << P << endl;
cout << "PmodI = " << PmodI << endl;
```

JAA confirms that PmodI has RingID = 4. Strange!

#5 - 27 Mar 2019 15:26 - John Abbott

Using gdb I see that **ComputeGBasis** is called, and it detected that the input ring is a polyring over a fractionfield (see line 129), and then it builds an internal polyring over ZZ for the GBasis computation (see line 132). This internal ring gets an ID of 3.

The GBasis computation was triggered by the test IsOne(I) at line 858 of QuotientRing.C.

Aha! Maybe Anna's impl has the clever trick that a non-zero principal ideal has an easy GBasis, so avoids calling ComputeGBasis... Clue city!

#6 - 27 Mar 2019 17:21 - Anna Maria Bigatti

- Status changed from New to Closed

- Assignee set to John Abbott

- % Done changed from 0 to 100

- Estimated time set to 2.01 h

John Abbott wrote:

> Using gdb I see that **ComputeGBasis** is called, and it detected that the input ring is a polyring over a fractionfield (see line 129), and then it builds an internal polyring over ZZ for the GBasis computation (see line 132). This internal ring gets an ID of 3.
>
> The GBasis computation was triggered by the test IsOne(I) at line 858 of QuotientRing.C.
>
> Aha! Maybe Anna's impl has the clever trick that a non-zero principal ideal has an easy GBasis, so avoids calling ComputeGBasis... Clue city!

true!!! wow, that was subtle....

#7 - 28 Mar 2019 16:36 - John Abbott

- Related to Feature #1249: principal ideal has a Gbasis added