**Function:** SchurFkt[AlexComp] - compares two compositions/partitions w.r.t. anti-lexicographic ordering

### Calling Sequence:

b := AlexComp(c1,c2)

### Parameters:

- c1,c2 : compositions (or partitions)

### Output:

- b : boolean value (true / false)

### WARNING:

Note that Maple uses in its combinatorial packages lexicographical order of partitions and compositions.

## Description:

- AlexComp allows to order compositions and partitions in anti-lexicographic order. Anti-lexicographic order is the standard order of  Macdonald and other writers on symmetric functions. Note that Maple uses in the combinatorics packages lexicographic order!

## Examples:

```
> restart:with(SchurFkt):
SchurFkt Version 1.0.2 (16 x 2008) at your service
(c) 2003-2008 BF&RA, no warranty, no fitness for anything!
Increase verbosity by infolevel[`function`]=val -- use online help > ?
Bigebra[help]
```

Check AlexComp for special values:

```
> AlexComp([3,2],[2,1]);
  AlexComp([2,2,1,0],[3,0,2,0]);
true
false
```

Using AlexComp to sort a list of partitions:

```
> prt:=[ [3,2,1,0,0,0],[6,0,0,0,0,0],[5,2,0,0,0,0],[1,1,1,1,1,
  1] ];
  sort(prt,AlexComp);
prt := [[3, 2, 1, 0, 0, 0], [6, 0, 0, 0, 0, 0], [5, 2, 0, 0, 0, 0], [1, 1,
1, 1, 1, 1]]
[[6, 0, 0, 0, 0, 0], [5, 2, 0, 0, 0, 0], [3, 2, 1, 0, 0, 0], [1, 1, 1, 1,
1, 1]]
```

Maples combinat produces lex ordered lists

```
> combinat[partition](3);
  sort(%,AlexComp);
[[1, 1, 1], [1, 2], [3]]
[[3], [1, 2], [1, 1, 1]]
```

Note that CompNM and PartNM functions produce lists of compositions and partitions in anti-lexcicographical order

```
> prt:=PartNM(3,3);
  cmp:=CompNM(2,5);
prt := [[3], [2, 1], [1, 1, 1]]
cmp := [[2, 0, 0, 0, 0], [1, 1, 0, 0, 0], [1, 0, 1, 0, 0], [1, 0, 0, 1,
0], [1, 0, 0, 0, 1], [0, 2, 0, 0, 0], [0, 1, 1, 0, 0], [0, 1, 0, 1, 0],
[0, 1, 0, 0, 1], [0, 0, 2, 0, 0], [0, 0, 1, 1, 0], [0, 0, 1, 0, 1], [0, 0,
0, 2, 0], [0, 0, 0, 1, 1], [0, 0, 0, 0, 2]]
```

Sorting in lexicographic  order my be achieved by using 'not'

```
> prt;
  sort(prt,not AlexComp);
  cmp;
  sort(cmp,not AlexComp);
[[3], [2, 1], [1, 1, 1]]
[[1, 1, 1], [2, 1], [3]]
[[2, 0, 0, 0, 0], [1, 1, 0, 0, 0], [1, 0, 1, 0, 0], [1, 0, 0, 1, 0], [1,
0, 0, 0, 1], [0, 2, 0, 0, 0], [0, 1, 1, 0, 0], [0, 1, 0, 1, 0], [0, 1, 0,
0, 1], [0, 0, 2, 0, 0], [0, 0, 1, 1, 0], [0, 0, 1, 0, 1], [0, 0, 0, 2, 0],
[0, 0, 0, 1, 1], [0, 0, 0, 0, 2]]
[[0, 0, 0, 0, 2], [0, 0, 0, 1, 1], [0, 0, 0, 2, 0], [0, 0, 1, 0, 1], [0,
0, 1, 1, 0], [0, 0, 2, 0, 0], [0, 1, 0, 0, 1], [0, 1, 0, 1, 0], [0, 1, 1,
0, 0], [0, 2, 0, 0, 0], [1, 0, 0, 0, 1], [1, 0, 0, 1, 0], [1, 0, 1, 0, 0],
[1, 1, 0, 0, 0], [2, 0, 0, 0, 0]]
```

AlexComp can handle lists with different length, if necessary trailing zeros are appended internally:

```
> AlexComp([2],[3,2,1,2]);
  AlexComp([4],[1,1,1,1]);
false
true
```

## Algorithm

Not available (obvious).

## See Also: SchurFkt[Overview], SchurFkt[PartNM], SchurFkt[CompNM]

Last modified: December 17, 2007/BF/RA.