```
------------------------------------------------------------------------------
   Issue )copyright to view copyright notices.
   Issue )summary for a summary of useful system commands.
   Issue )quit to leave AXIOM and return to shell.
Sunday August 31, 2003 at 07:30:21
------------------------------------------------------------------------------

(AXIOM Sockets) The AXIOM server number is undefined.


→ )show Integer

 Integer  is a domain constructor
 Abbreviation for Integer is INT
 This constructor is exposed in this frame.
 Issue )edit
/home/wspage/projects/axiom2/mnt/linux/../../src/algebra/INT.spad to see
algebra source code for INT

------------------------------- Operations --------------------------------
 ?*? : (%,%) -> %                        ?*? : (Integer,%) -> %
 ?*? : (PositiveInteger,%) -> %          ?**? : (%,PositiveInteger) -> %
 ?+? : (%,%) -> %                        ?-? : (%,%) -> %
 -? : % -> %                             ?<? : (%,%) -> Boolean
 ?<=? : (%,%) -> Boolean                 ?=? : (%,%) -> Boolean
 ?>? : (%,%) -> Boolean                  ?>=? : (%,%) -> Boolean
 D : % -> %                              D : (%,NonNegativeInteger) -> %
 OMwrite : (%,Boolean) -> String         OMwrite : % -> String
 1 : () -> %                             0 : () -> %
 ?^? : (%,PositiveInteger) -> %          abs : % -> %
 addmod : (%,%,%) -> %                   associates? : (%,%) -> Boolean
 base : () -> %                          binomial : (%,%) -> %
 bit? : (%,%) -> Boolean                 coerce : Integer -> %
 coerce : % -> %                         coerce : Integer -> %
 coerce : % -> OutputForm                convert : % -> String
 convert : % -> DoubleFloat              convert : % -> Float
 convert : % -> Pattern Integer          convert : % -> InputForm
 convert : % -> Integer                  copy : % -> %
 dec : % -> %                            differentiate : % -> %
 even? : % -> Boolean                    factor : % -> Factored %
 factorial : % -> %                      gcd : List % -> %
 gcd : (%,%) -> %                        hash : % -> %
 hash : % -> SingleInteger               inc : % -> %
 init : () -> %                          invmod : (%,%) -> %
 latex : % -> String                     lcm : List % -> %
 lcm : (%,%) -> %                        length : % -> %
 mask : % -> %                           max : (%,%) -> %
 min : (%,%) -> %                        mulmod : (%,%,%) -> %
 negative? : % -> Boolean                odd? : % -> Boolean
 one? : % -> Boolean                     permutation : (%,%) -> %
 positive? : % -> Boolean                positiveRemainder : (%,%) -> %
 powmod : (%,%,%) -> %                   prime? : % -> Boolean
 ?quo? : (%,%) -> %                      random : % -> %
 random : () -> %                        rational : % -> Fraction Integer
 rational? : % -> Boolean                recip : % -> Union(%,"failed")
 ?rem? : (%,%) -> %                      retract : % -> Integer
```

```
 sample : () -> %                          shift : (%,%) -> %
 sign : % -> Integer                       sizeLess? : (%,%) -> Boolean
 squareFree : % -> Factored %              squareFreePart : % -> %
 submod : (%,%,%) -> %                     symmetricRemainder : (%,%) -> %
 unit? : % -> Boolean                      unitCanonical : % -> %
 zero? : % -> Boolean                      ?~=? : (%,%) -> Boolean
 ?*? : (NonNegativeInteger,%) -> %
 ?**? : (%,NonNegativeInteger) -> %
 OMwrite : (OpenMathDevice,%,Boolean) -> Void
 OMwrite : (OpenMathDevice,%) -> Void
 ?^? : (%,NonNegativeInteger) -> %
 characteristic : () -> NonNegativeInteger
 differentiate : (%,NonNegativeInteger) -> %
 divide : (%,%) -> Record(quotient: %,remainder: %)
 euclideanSize : % -> NonNegativeInteger
 expressIdealMember : (List %,%) -> Union(List %,"failed")
 exquo : (%,%) -> Union(%,"failed")
 extendedEuclidean : (%,%) -> Record(coef1: %,coef2: %,generator: %)
 extendedEuclidean : (%,%,%) -> Union(Record(coef1: %,coef2: %),"failed")
 gcdPolynomial : (SparseUnivariatePolynomial %,SparseUnivariatePolynomial %)
-> SparseUnivariatePolynomial %
 multiEuclidean : (List %,%) -> Union(List %,"failed")
 nextItem : % -> Union(%,"failed")
 patternMatch : (%,Pattern Integer,PatternMatchResult(Integer,%)) ->
PatternMatchResult(Integer,%)
 principalIdeal : List % -> Record(coef: List %,generator: %)
 rationalIfCan : % -> Union(Fraction Integer,"failed")
 reducedSystem : Matrix % -> Matrix Integer
 reducedSystem : (Matrix %,Vector %) -> Record(mat: Matrix Integer,vec:
Vector Integer)
 retractIfCan : % -> Union(Integer,"failed")
 subtractIfCan : (%,%) -> Union(%,"failed")
 unitNormal : % -> Record(unit: %,canonical: %,associate: %)
```

```
→ integrate(sin(x),x)
```

$$-\cos(x) \tag{1}$$

Type: Union(Expression Integer,...)

```
→ integrate(x**(1/3),x)
```

$$\frac{3x\langle\text{root}|3\rangle\langle\text{of}|x\rangle}{4} \tag{2}$$

Type: Union(Expression Integer,...)

→