Complete the implementation of class `Matrix`, and further implement class `Vector` described below.

- Use `vector.h` and `matrix.h` to declare class `Vector` and `Matrix` respectively.
- Use `vector.cpp` and `matrix.cpp` to define the corresponding methods.
- Try to make as `const` methods as possible.
- Use `assert` to check if the operands are valid for an operator.

```
/**************************************************************************
* class Vector:
*
*   data member:
*     *entry      vector entries of type Double
*     size    vector size
*
*   constructors and destructor:
*     Vector(size)      create a vector object of size size
*     Vector(size,val)  create a vector with entries of val
*     Vector(v)         copy constructor from vector v
*     ~Vector()
*
*   member methods:
*     getSize()       return the size of vector v
*     setEntry(i,val) set v(i) = val
*     getEntry(i)     return v(i)
*     zeros()         initialize vector v with zero entries
*     ones()          initialize vector v with one entries
*     random()        initialize vector v with random entries
*     norm(p)         compute p-norm of vector v
*     print()         print the entries to the screen
*
*   entry access:
*     v(i)            zero-based indexing
*
*   member operators:
*     v = w         assignment
*     v = -w
*     ++v
*     v += w
*     v -= w
*     v *= alp        alp scalar number
*     v *= w
*     t = v + w
*     t = v - w
*     w = alp * v
**************************************************************************/
```

Listing 1: class Vector

```
 /**************************************************************************
* class Matrix:
*
*   member data:
*     **entry       matrix entries of type Double
```

```
*     row         number of rows
*     col         number of columns
*
*   constructors and destructor:
*     Matrix(m,n)     create an mxn matrix object
*     Matrix(m,n,val) create an mxn matrix with entries of val
*     Matrix(A)       copy constructor from matrix A
*     ~Matrix()
*
*   member methods:
*     getRows()           return number of rows
*     getCols()           return number of columns
*     setEntry(i,j,val)     set A(i,j) = val
*     getEntry(i,j)       return A(i,j)
*     zeros()               initialize matrix A with zero entries
*     ones()                initialize matrix A with one entries
*     eye()               initialize matrix A as an identity matrix
*     random()              initialize matrix A with random entries
*     print()               print the entries to the screen
*
*   entry access:
*     A(i,j)            i: row, j: column index
*
*   member operators:
*     A = B          assignment
*     ++A
*     A = -B
*     A += B
*     A *= alp        alp scalar number
*     A *= B
*     C = A + B
*     C = A - B
*     B = A * alp
*     C = A * B
*     B = transpose(A)
*************************************************************************/
```

Listing 2: class Matrix

- Use the attached `MatVecClass.cpp` to test your implementation.

```
#include <iostream>
#include "vector.h"         // header file for all functions relating to vector operations
#include "matrix.h"         // header file for all functions relating to matrix operations
using namespace std;

int main()
{

  const int p_INF = 100000;

  /*
  * 1. create vectors and matrices
  */
  Matrix A(5,4), B(5,4), C(5,4),
      D(4,5), E(5,5);
  Vector v(5), w(5), t(5);

  /*
  * 2. initialize vectors and matrices
  */
  v.random();
  w.random();
  t.zeros();
```

```
24
25    A.random();
26    B.random();
27    C.zeros();
28    D.random();
29    E.zeros();
30
31    /*
32    * 3. print out the initialized vectors and matrices
33    */
34    v.print();
35    w.print();
36    t.print();
37    A.print();
38    B.print();
39    C.print();
40    D.print();
41    E.print();
42
43
44    /*
45    * 4. compute vector norms
46    */
47    v.norm(2);
48    v.norm(p_INF);
49
50    /*
51    * 5. operations
52    */
53    //====== for vectors ========
54    //=== adding 2 vectors
55    t = v + w;
56    t.print();
57
58    //=== scalar-vector multiplication
59    t = v*alpha;
60    t.print();
61
62    //=== increament
63    ++t;
64    t.print();
65
66    //=== dot product of 2 vectors of the same size
67    beta = dot(v, w);
68    cout << "beta␣=␣" << beta << endl;
69
70    //====== for matrices ========
71    //==== adding 2 matrices
72    E.zeros();
73    C = A + B;
74    C.print();
75    E = C + B;  // should return error
76    //==== matrix-matrix multiplication
77    E = A * D;
78    E.print();
79    //==== matrix-scalar multiplication
80    E = E*alpha;
81    E.print();
82    //==== increament
83    ++E;
84    //==== matrix-vector multiplication
85    t = E * v;
86    t.print();
87
88    return 0;
89 }
```

Listing 3: MatVecClass.cpp