**Practical: 12 Derive a class ‘MAT’ from MATRIX class created in above program. Add a member function to overload ‘\*’ operator to multiply two objects. (Single Inheritance)**

#include "stdafx.h"

#include <iostream>

using namespace std;

class MATRIX

{ int arr[3][3];

public:

MATRIX operator +(MATRIX);

void indata(int x)

{ for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

arr[i][j] = x;

}

MATRIX()

{ indata(0); }

MATRIX(int y)

{ indata(y); }

void oudata()

{ cout << endl;

for (int i = 0; i < 3; i++)

{ for (int j = 0; j < 3; j++)

{ cout << arr[i][j] << "\t"; }

cout << endl;

}

}

int retdata(int x, int y)

{ return arr[x][y]; }

void putdata(int x,int y,int z)

{ arr[x][y] = z; }

};

MATRIX MATRIX :: operator +(MATRIX c1)

{ MATRIX c2;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

c2.arr[i][j] = arr[i][j] + c1.arr[i][j];

return c2;

}

class MAT : public MATRIX

{

public:

MAT operator \*(MAT);

MAT()

{ indata(0); }

MAT(int x)

{ indata(x); }

};

MAT MAT :: operator \*(MAT m1)

{ MAT m3;

int x;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

{ x = m1.retdata(0, j) \* retdata(i, 0);

x += m1.retdata(1, j) \* retdata(i, 1);

x += m1.retdata(2, j) \* retdata(i, 2);

m3.putdata(i, j, x);

}

return m3;

}

int main()

{ MAT m1(3), m2(6), m3;

cout << endl << "Matrix 1 : ";

m1.oudata();

cout << endl << "Matrix 2 : ";

m2.oudata();

m3 = m1 \* m2;

cout << endl << "Matrix 3 :";

m3.oudata();

return 0;

}

**Output 12**

