Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
CSE210: Object-Oriented Programming Language
Class Test-3, Marks-20, Time-25 Minutes

Q. Implement the member functions of the following C++ template class.

```
template<class T>
class matrix{
        int row;   //number of rows in the matrix
        int col;    //number of columns in the matrix
        T **mat;            //two dimensional array should be implemented with pointer and dynamically allocating memory
        public:
          matrix(int row, int col);                 //constructor of the generic matrix class
          ~matrix();                                //destructor of the generic matrix class
          matrix(const matrix &m);                  //copy constructor of the generic matrix class
          matrix operator +(matrix m2);             //+ operator overloaded for matrix addition
          matrix operator -(matrix m2);             //- operator overloaded for matrix subtraction
          matrix operator *(matrix m2);             //* operator overloaded for matrix multiplication
          matrix &operator=(matrix m2);             //= operator overloaded to safely assign one matrix to another matrix
          void show();                              //function to display the matrix elements in two dimensions
};
```

Answer:

```
#include <iostream>
#include <cstdlib>

using namespace std;

template<class T>
class matrix{
        int row;
        int col;
        T **mat;
        public:
                matrix(int row, int col);
                ~matrix();
                matrix(const matrix &m);
                matrix operator +(matrix m2);
                matrix operator -(matrix m2);
                matrix operator *(matrix m2);
                matrix &operator=(matrix m2);
                void show();
};

template<class T>
matrix<T>::matrix(int row, int col){
        int i,j;
        this->row=row;
        this->col=col;
        mat=new T*[row];

        for(i=0; i<row; i++){
                mat[i]=new T[col];

        }
        for(i=0;i<row;i++){
                for(j=0; j<col; j++){
                        mat[i][j]=1;
```

```
                }
        }

}

template<class T>
matrix<T>::~matrix(){
        delete []mat;
}

template<class T>
matrix<T>::matrix(const matrix &m){
        int i,j;
        row=m.row;
        col=m.col;
        mat=new T*[row];
        for(i=0;i<row; i++){
                mat[i]=new T[col];
        }
        for(i=0;i<row;i++){
                for(j=0;j<col; j++){
                        mat[i][j]=m.mat[i][j];
                }
        }

}

template<class T>
matrix<T> matrix<T>::operator+(matrix m2){
        matrix<T> temp(row,col);
        int i, j;
        for(i=0;i<row; i++){
                for(j=0; j<col; j++){
                        temp.mat[i][j]=mat[i][j]+m2.mat[i][j];
                }
        }
        return temp;
}

template<class T>
matrix<T> matrix<T>::operator - (matrix m2){
        matrix<T> temp(row, col);
        int i, j;
        for(i=0;i<row;i++){
                for(j=0;j<col;j++){
                        temp.mat[i][j]=mat[i][j]-m2.mat[i][j];
                }
        }
        return temp;
}

template<class T>
matrix<T> matrix<T>::operator *(matrix m2){
        matrix<T> temp(row, col);
        int i,j, l;
        for(i=0;i<row;i++){
                for(j=0;j<col;j++){
                        temp.mat[i][j]=0;
                        for(l=0;l<col;l++){
                                temp.mat[i][j]+=mat[i][l]*m2.mat[l][i];
                        }
                }
```

```
        }
        return temp;
}

template<class T>
matrix<T> &matrix<T>::operator = (matrix m2){
        int i,j;
        for(i=0; i<row; i++){
                for(j=0;j<col; j++){
                        mat[i][j]=m2.mat[i][j];
                }
        }
        return *this;
}

template<class T>
void matrix<T>::show(){
        for(int i=0;i<row;i++){
                for(int j=0;j<col; j++){
                        cout<<mat[i][j]<<"\t";
                }
                cout<<endl;
        }
}

void matrix<double>::show(){
        for(int i=0;i<row;i++){
                for(int j=0;j<col; j++){
                        printf("%lf\t",mat[i][j]);
                }
                cout<<endl;
        }
}


int main(){
        matrix<int> im1(3,3), im2(3,3), im3(3,3);
        im1.show();
        im2.show();
        im3=im1+im2;
        im3.show();
        im3=im1-im2;
        im3.show();
        im3=im1*im2;
        im3.show();
        matrix<double> dm1(4,4), dm2(4,4), dm3(4,4);
        dm1.show();
        dm3=dm1+dm2;
        dm3.show();
        dm3=dm1-dm2;
        dm3.show();
        dm3=dm1*dm2;
        dm3.show();

        return 0;
}
```