Dynamic Memory Allocation of Array in C++

```
#include<iostream>
using namespace std;

class OneDimentional
{
	private:
		int iSize;
		int *p;
	public:
		OneDimentional(int);
		OneDimentional(OneDimentional &);
		~OneDimentional();
		void Accept();
		void Display();
		int Addition();
		int Maximum();
};

OneDiemntional::OneDiemntional(int iLength = 10)
{
	iSize = iLength;
	p = new int[iSize];
}

OneDiemntional::OneDiemntional(OneDiemntional &ref)
{
	this->iSize = ref.iSize;
	this->p = new int[iSize];

	for(int i = 0 ; i < iSize ; i++)
	{
		this->p[i] = ref.p[i];
	}
}

OneDiemntional::~OneDiemntional()
```

```
{
        delete []p;
}

void OneDiemntional::Accept()
{
        cout<<"Enter the elements\n";

        for(int i = 0 ; i < iSize ; i++)
        {
                cin>>p[i];
        }
}

void OneDiemntional::Display()
{
        cout<<"Elemets in 1D array are\n";

        for(int i = 0 ; i < iSize ; i++)
        {
                cout<<p[i];
        }
}

int OneDiemntional::Addition()
{
        int iSum = 0;

        for(int i = 0 ; i < iSize ; i++)
        {
                iSum = iSum + p[i];
        }
}

int OneDiemntional::Maximum()
{
        int iMax = p[0];
        for(int i = 0 ; i < iSize ; i++)
        {
                if(p[i] > iMax)
                        iMax = p[i];
        }

        return iMax;
}

class TwoDimentional
{
        private:
                int iRow,iCol;
                int **p;
        public:
                TwoDimentional(int,int);
                TwoDimentional(OneDimentional &);
                ~TwoDimentional();
                void Accept();
                void Display();
};


TwoDimentional::TwoDimentional(int iValue1 = 4, int iValue2 = 4)
{
        iRow = iValue1;
        iCol = iValue2

        p = new int*[row];


        for(int i = 0 ; i < iRow ; i++)
        {
                p[i] = new int[iCol];
        }
}

TwoDimentional::TwoDimentional(OneDiemntional &ref)
```

```
{
        this->iRow = ref.iRow;
        this->iCol = ref.iCol;
        this->p = new int*[iRow];

        for(int i = 0 ; i < iRow ; i++)
        {
                this->p[i] = new int[iCol];
        }

        for(int i = 0 ; i < iRow ; i++)
        {
                for(int j = 0 ; j < iCol ; j++)
                {
                                this->p[i][j] = ref.p[i][j];
                }
        }
}

TwoDimentional::~TwoDimentional()
{
        for(i = 0 ; i < row ; i++)
        {
                delete []p[i];
        }
        delete []p;
}

void TwoDimentional::Accept()
{
        cout<<"Enter elemets in 2D array\n";

        for(int i = 0 ; i < iRow ; i++)
        {
                for(int j = 0 ; j < iCol ; j++)
                {
                                cin>>p[i][j];
                }
        }
}

void TwoDimentional::Display()
{
        cout<<"Elemets in 2D array are\n";

        for(int i = 0;i < iRow ; i++)
        {
                cout<<"\n";
                for(int j = 0 ; j < iCol ; j++)
                {
                                cout<<p[i][j];
                }
        }
}

int main()
{
        OneDiemntional oobj1(12);
        oobj1.Accept();
        oobj1.Display();

        int iRet = 0;
        iRet = oobj1.Addition();
        cout<<"Addition is"<<iRet;
        iRet = oobj1.Maximum();
        cout<<"Maximum element is"<<iRet;


        OneDiemntional oobj2(oobj1);
        oobj2.Display();

        OneDiemntional *optr = new OneDiemntional(20);
        optr->Accept();
        optr->Display();
        delete optr;
```

```
	TwoDimentional tobj1(3,5);
	tobj1.Accept();
	tobj1.Display();

	TwoDimentional tobj2(tobj1);
	tobj2.Display();

	TwoDimentional tobj3();
	tobj3.Accept();
	tobj3.Display();

	TwoDimentional tobj4(3);
	tobj4.Accept();
	tobj4.Display();


	TwoDiemntional *tptr = new OneDiemntional(4,5);
	tptr->Accept();
	tptr->Display();
	delete tptr;

	return 0;
}
```