



### Object Oriented Programming 1<sup>st</sup> Midterm Examination Solutions

#### Question1:

- a) Copy constructor must copy the memory locations pointed by p, not the value of the pointers.  
This class needs an assignment operator to copy memory locations pointed by p.  
b) c) and d) Try by yourselves.

#### Question2:

a)

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

class Polynomial{
    int degree;           // degree
    int *coeff;           // pointer to coefficients
public:
    Polynomial(int , const int *);           // Constructor
    Polynomial(const Polynomial &);         // Copy constructor
    ~Polynomial(){ delete[] coeff; } // Destructor
    const Polynomial & operator=(const Polynomial &); // Assignment
    Polynomial operator+(const Polynomial &) const; // +
    char operator==(const Polynomial &) const; // ==
    long operator()(int) const; // (int)
    double operator()(float) const; // (float)
    int & operator[](int); // []
};

Polynomial::Polynomial(int deg, const int *co=0) // Constructor
{
    if (deg >0){
        degree=deg;
        coeff=new int[degree+1];
        int count;
        for (count=0; count <=degree; count++) coeff[count]=1;
        if (co)
            for (count=0; count <=degree; count++) coeff[count]=co[count];
    }
    else
        cerr << " ERROR: Degree" << endl;
}

Polynomial::Polynomial(const Polynomial &in) // Copy constructor
{
    degree=in.degree;
    coeff=new int[degree+1];
    int count;
    for (count=0; count <=degree; count++)
        coeff[count]=in.coeff[count];
}
```

```

const Polynomial & Polynomial::operator=(const Polynomial &in) // Assignment
{
    degree=in.degree;
    delete coeff;
    coeff=new int[degree+1];
    int count;
    for (count=0; count <=degree; count++)
        coeff[count]=in.coeff[count];
    return *this;
}

Polynomial Polynomial::operator+(const Polynomial &in) const // +
{
    int maxdeg,mindeg;
    int *cf;
    maxdeg = degree > in.degree ? degree : in.degree;
    mindeg = degree < in.degree ? degree : in.degree;
    cf=new int[maxdeg];
    int count;
    for (count=0; count <=mindeg; count++)
        cf[count] = coeff[count] + in.coeff[count];
    for (count=mindeg+1; count<=maxdeg; count++)
        if(maxdeg==degree) cf[count]=coeff[count];
        else cf[count]=in.coeff[count];
    return Polynomial(maxdeg,cf);
};

char Polynomial::operator==(const Polynomial &in) const // ==
{
    if (degree != in.degree) return 0;
    for (int count=0; count <=degree; count++)
        if (coeff[count]!=in.coeff[count]) return 0;
    return 1;
}

long Polynomial::operator()(int x) const // (int)
{
    long val=0L;
    for (int count=0; count <=degree; count++)
        val += pow(x,count)*coeff[count];
    return val;
}

double Polynomial::operator()(float x) const // (float)
{
    double val=0.0;
    for (int count=0; count <=degree; count++)
        val += pow(x,count)*coeff[count];
    return val;
}

int & Polynomial::operator[](int i) // []
{
    if (i >=0 && i<=degree)
        return coeff[i];
    else{
        cerr << "Out of bounds" << endl;
        return coeff[0];
    }
}

```

**b)**

```
void main()
{
    int i1[]={1,2,0,4,8};
    int i2[]={5,0,4,7,};
    Polynomial p1(4,i1),p2(3,i2),p3(1);
    p3=p1+p2;
    Polynomial p4=p1;
    if (p1==p4) cout << "They are equal" << endl;
    else cout << "They are not equal" << endl;
    p1[2]=18;
    cout << "2nd coefficient of p1= " << p1[2] << endl;
    cout << "value of p2(3)= " << p2(3) << endl;
    cout << "value of p3(1.4)= " << p2((float)1.4) << endl;
}
```