1.

According to the class note

An iterative algorithm to multiply two polynomials.

```
FOR I = 0 TO n-1
   FOR J = 0 TO n-1
      C[I + J] := A[I] * B[J] + C[I + J]
   ENDFOR
ENDFOR
```

```c
int main() {
        int ordP1,ordP2;
        int Poly1[100],Poly2[100],Poly3[100]={0};
        printf("the max power of the polynomial : ");
        scanf("%d",&ordP1);
        printf("coefficient of the polynomial 1:(from x^0 to x^n) ");
        for(int i=0;i<=ordP1;i++)
            scanf("%d",&Poly1[i]);
        printf("the max power of the polynomial 2: ");
        scanf("%d",&ordP2);
        printf("coefficient of the polynomial 2:(from x^0 to x^n) ");
        for(int i=0;i<=ordP2;i++)
            scanf("%d",&Poly2[i]);

        for(int i=0;i<=ordP1;i++){
            for(int j=0;j<=ordP2;j++){
                Poly3[i+j]+=Poly1[i]*Poly2[j];
            }
        }

        //output the polynomial 3
        for(int i=ordP1+ordP2;i>=0;i--)
            printf("%d ",Poly3[i]);
        printf("\n");
        return 0;
}
```

2.According to the website

https://www.geeksforgeeks.org/multiply-two-polynomials-2/

https://stackoverflow.com/questions/13497810/finding-an-error-in-code-for-divide-and-conquer-polynomial-multiplication

http://www.cs.ust.hk/mjg_lib/Classes/COMP3711H_Fall14/lectures/DandC_Multiplication_Handout.pdf

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#define MAX(a,b) (((a)>(b))?(a):(b))
int *getHalfList(int *x,char a,int l){
    int rt[l/2];
    assert(rt);
    if ( a == 'L')
        for (int i = 0; i < l/2 ;i++)
            rt[i]=x[i];
    else
        for (int i = l/ 2; i < l; i++)
        {
            rt[i]=x[i];
        }
    return rt;                                  ⚠ Address of stack memory associated with l
}
int *mergeList(int *p1, int *p2,int *p3,int len1,int len2, int n){
    int* rt = malloc(sizeof(int)* len1+len2-1);
    for (int i = 0; i < n/2; i++){
        if (i < len1)
            rt[i] += p1[i];

        if (i >= n / 2 && i < (n / 2 + len2))
            rt[i] += p2[i - n / 2];

        if (i >= n)
            rt[i] += p3[i - n];
    }
    return rt;
}
int *addList(int *a,int *b,int n){
    int* rt = malloc(sizeof(int)* n/2);
    for (int i = 0; i < n/2; i++)
        rt[i]=a[i]+b[i];
    return rt;
}
int *minusList(int *p3,int *p1,int *p2,int n){
    int* rt = malloc(sizeof(int)* n/2);
    for (int i = 0; i < n/2; i++)
        rt[i]=p3[i]-p1[i]-p2[i];
    return rt;
```

```c
int *multiply(int *Poly1,int *Poly2){
    int i=0,k=0;
    while(Poly1[i]){i++;}
    while(Poly2[k]){k++;}
    int n;
    if(i>k){n=i;}else{n=k;}

    if(n == 1){
        int* a = malloc(sizeof(int)* 1);
        a[0]=Poly1[0] * Poly2[0];
        return a;
    }
    else if (n==0){
        return 0;
    }
    else {
        int *p0 = getHalfList(Poly1,'L',i);
        int *p1 = getHalfList(Poly1,'R',i);
        int *q0 = getHalfList(Poly2,'L',k);
        int *q1 = getHalfList(Poly2,'R',k);

        int *x1 = multiply(p0, q0);
        int *x2 = multiply(p1,q1);
        int *x3 = multiply(addList(p0, q0,n), addList(p1, q1,n));
        return mergeList(p1, minusList(x3,x1,x2,n), x2,i,k, n);
    }
}
```

```c
int main(int argc, const char * argv[]) {
    int ordP1,ordP2;
    int Poly1[100],Poly2[100]={0};
    printf("the max power of the polynomial : ");
    scanf("%d",&ordP1);
    printf("coefficient of the polynomial 1:(from x^0 to x^n) ");
    for(int i=0;i<=ordP1;i++)
        scanf("%d",&Poly1[i]);
    printf("the max power of the polynomial 2: ");
    scanf("%d",&ordP2);
    printf("coefficient of the polynomial 2:(from x^0 to x^n) ");
    for(int i=0;i<=ordP2;i++)
        scanf("%d",&Poly2[i]);
    printf("start");


    int *Poly3=multiply(Poly1,Poly2);
    for(int i=0;i<ordP1+ordP2;i++){
        printf("%d",Poly3[i]);
    }
    printf("\n");

    return 0;
}
```

3.According to this two website

https://hk.saowen.com/a/b6e9f0ca70a669575a8b8e56c746ba63780958c4c12159aec6bfb05a7ff2e409

http://www.voidcn.com/article/p-rzrkhina-boa.html

https://activities.tjhsst.edu/sct/lectures/1415/SCT_Multiplying_Polynomials.pdf

main

```
    for(int i=1; i<=abs(ordP1-ordP2) && (ordP1-ordP2!=0) ;i++){

        if(ordP1-ordP2<0){
            Poly1[i+ordP1]*=Complex(0.0,0.0);

        }else{
            Poly2[i+ordP2]*=Complex(0.0,0.0);
        }
    }

    len = (ordP1-ordP2)?(ordP1+1):(ordP2+1);

    for(len = 2; len < ordP1+ordP2+2;len*=2);

    std::cout<<len<<"\\\\"<<'\n';
    for(int i =ordP1+1;i<len;i++){
        Poly1[i] = Poly1[i] * Complex(0.0,0.0);
        Poly2[i] = Poly2[i] * Complex(0.0,0.0);
    }


    FFT(Poly1,len,1);
    FFT(Poly2,len,1);

    for (int i =0;i<len;i++){
        Poly3[i]=Poly1[i]*Poly2[i];
    }

    FFT(Poly3,len,-1);
    for (int i =0;i<len;i++){
        Poly3[i].r /= len;
    }
```

```
struct Complex
{
    double r, i;
    Complex() {}
    Complex(double _r, double _i) { r = _r; i = _i; }
    Complex operator +(const Complex &y) { return Complex(r + y.r, i + y.i); }
    Complex operator -(const Complex &y) { return Complex(r - y.r, i - y.i); }
    Complex operator *(const Complex &y) { return Complex(r*y.r - i * y.i, r*y.i + i * y.r); }
    Complex operator *=(const Complex &y) {
        double t = r;
        return Complex(r = r * y.r - i * y.i, i = t * y.i + i * y.r); }  ⚠ Unsequenced modification and
} a[MAXN], b[MAXN];
```

```
void FFT(Complex* a,long len,int op){

    if(len==1)
        return;
    Complex * a0=new Complex[len/2];
    Complex * a1=new Complex[len/2];
    for(long i=0;i<len;i+=2){
        a0[i/2]=a[i];
        a1[i/2]=a[i+1];
    }

    FFT(a0,len/2,op);
    FFT(a1,len/2,op);
    Complex wn(cos(2*Pi/len),op*sin(2*Pi/len));
    Complex w(1,0);

    for(long i=0;i<(len/2);i++){
        a[i]=a0[i]+w*a1[i];
        a[i+len/2]=a0[i]-w*a1[i];
        w=w*wn;

    }

    delete[] a0;
    delete[] a1;
}
```

4. I think the FFT is the fastest one of the three methods.
iterative method

An iterative algorithm to multiply two polynomials.

```
FOR I = 0 TO n-1
   FOR J = 0 TO n-1
      C[I + J] := A[I] * B[J] + C[I + J]
   ENDFOR
ENDFOR
```

According to the classnote

$P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$
$Q(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}$

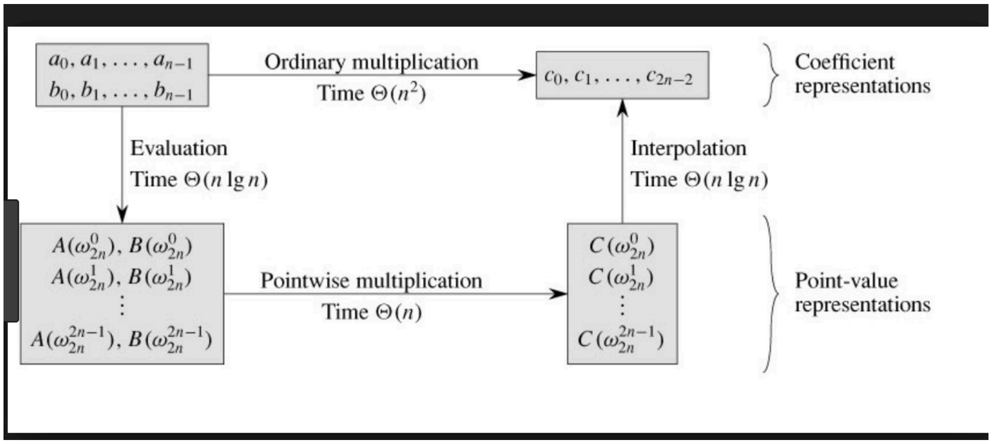
$P(x)Q(x) = c_0 + c_1x + \ldots + c_{2n-2}x^{2n-2}$, where Cn=Sigma(ak*b(i-k))
every a_i is multiply with every $b_j \Longrightarrow \Theta(n^2)$

divide and conquer method
According to https://www.geeksforgeeks.org/multiply-two-polynomials-2/
$P(x) = P_0(x) + P_1(x)x^{n/2}$
$Q(x) = Q_0(x) + Q_1(x)x^{n/2}$
Then
$P(x)Q(x)$
$= P_0(x)Q_0(x) + (P_0(x)\, Q_1(x) + P_1(x)Q_0(x))x^{n/2} + P_1(x)Q_1(x)x^n$
It is only uses 3 half size multiplications
$M(n) = 3M(n/2) + cn = 3^k(n/2^k) + (1+2+4+\ldots+2^{k-1})cn = 3^{\log n} M(1) + cn(2^{\log n} - 1) = 3^{\log n} M(1) + cn(n - 1) \Longrightarrow \Theta(3^{\log n}) \Longrightarrow \Theta(n^{\log 3})$

FFT method
According to the website: http://web.cs.iastate.edu/~cs577/handouts/polymultiply.pdf

http://blog.csdn.net/v_july_v/article/details/6684636

$$\overline{X} \xrightarrow{\text{PERMUTE}} X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \xrightarrow{\text{HALF-SIZE}} \text{F} \quad \begin{pmatrix} F_nX_1 \\ F_nX_2 \end{pmatrix} \xrightarrow{\text{COMBINE}} \begin{pmatrix} F_nX_1 + DF_nX_2 \\ F_nX_1 - DF_nX_2 \end{pmatrix}$$



The first part is Θ(nlogn)

The second part is Θ(n)

The third part is Θ(nlogn)

Therefore, the total is Θ(nlogn+n+ nlogn)= Θ(nlogn)

5.

First time:3x^3+x^2+2x+2 and 3x^3+2x^2+x+3

iterative method: `9 12 13 21 12 8 6`

Divide & conquer: `9 12 13 21 12 8 6`

FFT:
```
9.000000 12.000000 13.000000 21.000000 12.000000 8.000000
6.000000
```

Second time:x^3+2x^2+3x+3 and x^3-x^2-4x-2 iterative method:

```
the max power of the polynomia : 3
coefficient of the polynomial 1:(from x^0 to x^n) 3
3
2
1
the max power of the polynomial 2: 3
coefficient of the polynomi16al 2:(from x^0 to x^n)
-2
-4
-1
1
1 1 -3 -10 -19 -18 -6 0.000003 seconds

Program ended with exit code: 0
```

Divide & conquer:

```
the max power of the polynomia : 3
coefficient of the polynomial 1:(from x^0 to x^n) 3
3
2
1
the max power of the polynomial 2: 3
coefficient of the polynomi16al 2:(from x^0 to x^n)
-2
-4
-1
1
1 1 -3 -10 -19 -18 -6 0.000003 seconds

Program ended with exit code: 0
```

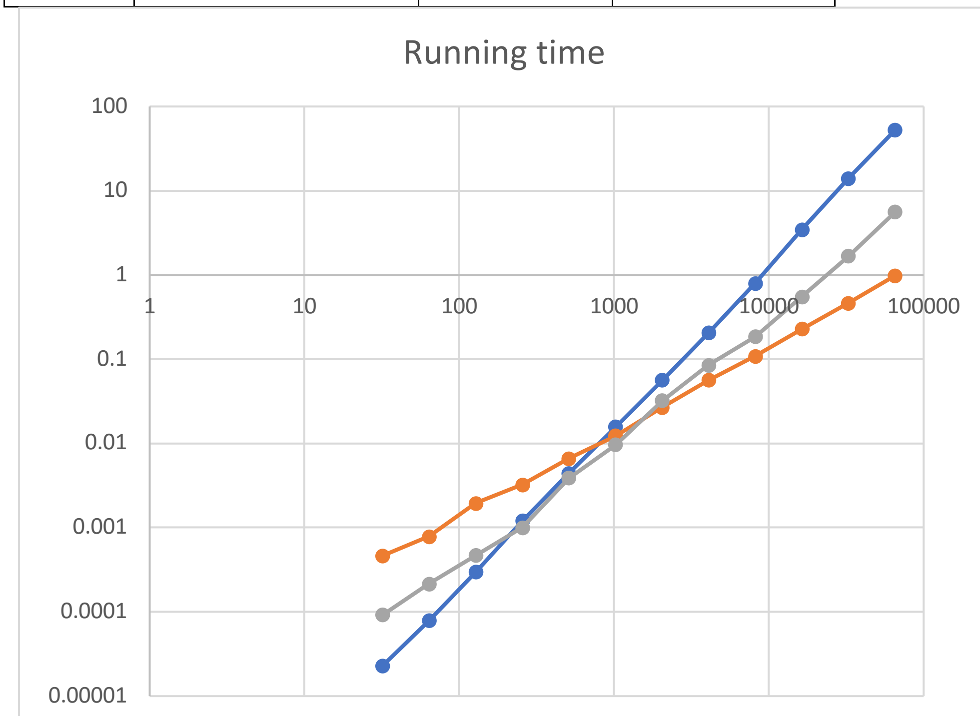
FFT:

From those two tests, we can see all the answer are same.

6.

n	iterative (sec)	DE (sec)	FFT (sec)
32	0.000023	0.000092	0.000465
64	0.000079	0.000215	0.000789
128	0.000298	0.000468	0.001951
256	0.001201	0.001002	0.003231
512	0.00445	0.003876	0.006653
1024	0.015724	0.009625	0.012363
2048	0.056513	0.032252	0.026828

4096	0.205633	0.085493	0.056278
8192	0.795653	0.18566	0.108544
16384	3.452353	0.554307	0.228919
32768	14.053701	1.682307	0.465416
65536	52.652832	5.588962	0.983676



7.According to the plot in Q6

Iterative: Θ(n^2) the mean coefficients is 1.55024E-08

DE: Θ(n^log3) the mean coefficients is 1.76055E-08

FFT: Θ(n*logn) the mean coefficients is 4.87361E-08

The crossover point of Iterative and DE: n=1512

At the beginning the Iterative is fast than DE, but after the point 1512, it will change, the DE will fast than Iterative

This is same as the plot and dataset.

The crossover point of FFT and DE: n=824

At the beginning the Iterative is fast than FFT, but after the point 824, it will change, the FFT will fast than Iterative

This is same as the plot and dataset.