유한체 상에서 정의된 기약다항식의 판별에 관한 연구

Jang-Hyuck-Choi
Kookmin University
jhdh0813@kookmin.ac.kr

Updated: 2020년 11월 18일

차 례

제 1 장

목표

1. 유한체 상에서 정의된 다항식의 기약성 판별.
2. 유한체 상에서 정의된 기약 다항식 임의 생성.

제 2 장

Prime Field

제 1 절 Ring Homomorphism

Theorem 2.1.1 <Ring homomorphism>
If R is a ring with unity 1, then the map $\phi : Z \to R$ given by $\phi(n) = n * 1$ for $n \in Z$ is a ring homomorphism of Z into R.

위 Theorem은
$\phi(m + n) = \phi(m) + \phi(n)$ 과
$\phi(m * n) = \phi(m) + \phi(n)$ 을 보임으로써 쉽게 증명 할 수 있다.

Corollary 2.1.2
-If ring R has char 0, R has a subring isomorphic to Z.
-If R has char $n \geq 1$, R has a subring isomorphic to Z_n.

위의 따름정리를 통해 다음을 얻을 수 있다.

Corollary 2.1.3
-If field $\mathbb{F}$ has char 0, $\mathbb{F}$ has a subfield isomorphic to $\mathbb{Q}$.
-If field $\mathbb{F}$ has prime char p, $\mathbb{F}$ has a subfield isomorphic to Z_p.

제 2 절 Principle Ideal Domain

- Principle Ideal Domain(PID)은 모든 아이디얼이 생성자가 단 1개인 것을 의미한다.
- $\forall I \triangleleft \mathbb{Z}, \exists n \in Z$, such that $I = (n) = nZ$
- 따라서 정수는 생성자 n에 의한 PID이다.
- 그렇다면 $\mathbb{F}[x]$의 PID 여부를 확인해야 한다.

Theorem 2.2.1
If $\mathbb{F}$ is a field, every ideal in $\mathbb{F}[x]$ is principle.

Pf) Let $N \triangleleft \mathbb{F}[x]$

1. $N =< 0 >$
2. $N =< 1 >$
3. $N =< g(x) >$

위 세가지 경우 모두 principle임을 보임으로써 증명 할 수 있다.

Theorem 2.2.2
An ideal $< p(x) > \neq \{0\}$ in $\mathbb{F}[x]$ is maximal ideal if and only if $p(x)$ is irreducible over $\mathbb{F}$.

Pf)

($\Rightarrow$) Suppose $< p(x) >$ is maximal in $\mathbb{F}[x]$. Then $< p(x) >\subset \mathbb{F}[x]$.

If $p(x) = f(x) * g(x)$, then $< p(x) > \ = \ < f(x) * g(x) >$

$\rightarrow \ f(x) * g(x) \ \in < p(x) >$

$\rightarrow f(x) \ \in < p(x) >$ or $g(x) \ \in < p(x) >$

Then $\exists t(x) \in \mathbb{F}[x]$ such that $f(x) = p(x) * t(x)$ **or** $\exists s(x) \in \mathbb{F}[x]$ such that $g(x) = p(x) * s(x)$.

$\rightarrow$ deg $p(x) \leq$ deg $f(x)$ or deg $p(x) \leq$ deg $g(x)$

이는 $p(x) = f(x) * g(x)$ 에 모순이다.

$\therefore p(x)$ is irreducible over $\mathbb{F}$.

($\Leftarrow$) If $p(x)$ is irreducible over $\mathbb{F}$, then $< p(x) >$ is maximal in $\mathbb{F}[x]$.

Then $< p(x) >\neq \mathbb{F}[x]$ since $p(x)$ is irreducible over $\mathbb{F}$.

If $\exists N \triangleleft \mathbb{F}[x]$ such that $< p(x) > \leq \ N \ \leq \ \mathbb{F}[x]$, since $\mathbb{F}[x]$ is PID, $\exists g(x) \in \mathbb{F}[x]$ such that $N =< g(x) >$.

즉 $< p(x) > \leq \ g(x) \ \leq < 1 >$ 이다.

$\rightarrow \exists h(x) \in \mathbb{F}[x]$ such that $p(x) = g(x) * h(x)$.

Since $p(x)$ is irreducible over $\mathbb{F}$, we get either $g(x) \in \mathbb{F}$ or $h(x) \in \mathbb{F}$.

In first case, $< g(x) >= N = \mathbb{F}[x]$,

and in second case, $h(x) = c \in \mathbb{F}$ with $c \neq 0$.

$\rightarrow p(x) = c * g(x)$.

$\therefore g(x) = c^{-1} * p(x)$.

결국 $g(x)$또한 $p(x)$의 배수꼴이 된다.

$\rightarrow g(x) \in < p(x) >$.

$\rightarrow N =< g(x) > \subset < p(x) >$

$\therefore N =< p(x) >$

위 Theorem으로 다음과 같은 정리를 얻을 수 있다.

Theorem 2.2.3

$\mathbb{F}[x]$ 는 ring이고 $< p(x) >$ 는 maximal ideal 이므로 $\mathbb{F}[x]$ / $< p(x) >$ 는 Field 이다.

Ex) $x^3 + 3x + 2 \in \mathbb{Z}_5[x]$ is irreducible over $\mathbb{Z}_5$

Check $\mathbb{Z}_5[x]$ / $< x^3 + 3x + 2 >$ is field.

제 3 장

Extenstion of Field

제 1 절 Kronecker's Theorem

Theorem 3.1.1 <Kronecker's Theorem>
Let $\mathbb{F}$ be a field and $f(x)$ be a non-constant polynomial in $\mathbb{F}[x]$(with deg $f(x) \geq 1$).
Then there exists an extenstion field $\mathbb{E}$ of $\mathbb{F}$ and an $\alpha \in \mathbb{E}$ such that $f(\alpha) = 0$.

제 2 절 Algebraic

Definition 3.2.2
An element α of an extension field $\mathbb{E}$ of a field $\mathbb{F}$ is algebraic over $\mathbb{F}$(= 대수적이다) if $f(\alpha) = 0$ for some non-zero $f(x) \in \mathbb{F}[x]$. If α is not algebraic over $\mathbb{F}$, then α is transcendental over $\mathbb{F}$(= 초월적이다).

Ex) $\mathbb{Q} \leq \mathbb{R}, \ \sqrt{2} \in \mathbb{R}, \ f(x) = x^2 - 2 \ \in \ \mathbb{Q}[x] \rightarrow \ f(\sqrt{2}) = 0$
$\therefore \sqrt{2}$ is algebraic over $\mathbb{F}$.

Theorem 3.2.3

$\mathbb{F} \leq \mathbb{E}$: field extension, $\alpha \in \mathbb{E}$, α is algebraic over $\mathbb{F}$.

Then

1. There exists an irreducible polynomial $p(x) \in \mathbb{F}[x]$ such that $p(\alpha) = 0$.
2. And this irreducible polynomial $p(x)$ is uniquely determined up to a constant factor in $\mathbb{F}[x]$.
3. And if $f(\alpha) = 0$ for $f(x) \in \mathbb{F}[x]$, then $p(x)$ divides $f(x)$

또한 이러한 기약다항식을 $p(x) = \mathrm{irr}(\alpha, \mathbb{F})$로 정의한다.

Pf)

Let $\phi_\alpha : \mathbb{F}[x] \to \mathbb{E}$

$\Rightarrow \ker \phi_\alpha \triangleleft \mathbb{F}[x]$

$\Rightarrow \exists\ p(x) \in \mathbb{F}[x]$, $\ker \phi_\alpha = < p(x) >$ such that $p(\alpha) = 0$.(since $\mathbb{F}[x]$ is PID)

If $f(\alpha) = 0$ and $f(x) \neq$ in $\mathbb{F}[x]$

then $\phi_\alpha(f(x)) = f(\alpha) = 0$.

$f(x) \in < p(x) > \Rightarrow\ \ g(x) \in \mathbb{F}[x]$ such that $f(x) = p(x) * g(x)$.

$\therefore p(x)$ divides $f(x)$

$p(x)$보다 더 낮은 차수가 존재 할 수 없다.

$\therefore p(x)$ has a minimal degree among all the polynomial $f(x)$ such that $f(\alpha) = 0$

p(x)의 기약성도 확인해야 한다.

If $p(x) = r(x) * s(x)$ of lower degrees, (즉 $r(x) \in \mathbb{F}[x]$, $s(x) \in F[x]$)

p$(\alpha) = 0 = r(\alpha) * s(\alpha) \in \mathbb{E}$.

$\to r(\alpha) = 0$ or $s(\alpha) = 0$

$\to p(x)$가 minimal 임에 모순이다.

$\therefore p(x)$ is irreducible

제 3 절 Simple Extension

Definition 3.3.1

$\mathbb{F} \leq \mathbb{E}$, $\mathbb{E}$ is simple extension of $\mathbb{F}$.

$\to$ Then there exist $\alpha \in \mathbb{E}$ such that $\mathbb{E} = \mathbb{F}(\alpha)$.

Ex) $\mathbb{Q}(\sqrt{2})$ 는 $\mathbb{Q}$의 단순확대체이다.

Theorem 3.3.2

If $\mathbb{E} = \mathbb{F}(\alpha)$ where α is algebraic over $\mathbb{F}$ with $\deg[\mathrm{irr}(\alpha, \mathbb{F})] = n \geq 1$.

Then $\forall \beta \in \mathbb{E} = \mathbb{F}(\alpha)$, there uniquely exists $b_0, b_1, \ldots, b_{n-1} \in \mathbb{F}$ such that

$\beta = b_0 * 1 + b_0 * \alpha^1 + \ldots + b_{n-1} * \alpha^{n-1}$

위 theorem이 증명된다면 이제 이 단순확장체는 스칼라 $\mathbb{F}$와 $\mathbb{F}$의 기저 n개로 이루어진 벡터공간이 된다.

제 4 절 Finite Extension

Definition 3.4.1

체 $\mathbb{F}$의 확대체 $\mathbb{E}$가 n차원 $\mathbb{F}$-벡터공간이 된다면 $\mathbb{F}$는 차수가 n인 $\mathbb{F}$의 유한확대체(finite extension)라고 한다.

이때의 차수 n을 $[\mathbb{E} : \mathbb{F}]$로 적는다.

위 정의서 주의해야 할 점은 유한체라는 것이 아니라 기저의 개수가 유한개임을 의미한다는 것이다.

Theorem 3.4.2

A finite extension E of F is an algebraic extension of F.

체 $\mathbb{F}$의 유한확대체 $\mathbb{E}$는 $\mathbb{F}$의 대수적 확대체이다.
즉, 기저가 유한개인 확장(extension)이면 임의의 $\alpha \in \mathbb{E}$는 $\mathbb{F}$에 대수적이다.

Pf) For all $\alpha \in \mathbb{E}$, $[\mathbb{E}:\mathbb{F}] = n < \infty$

$\{1, \alpha, \alpha^2, \ldots, \alpha^n\} \subset \mathbb{E}$: linearly dependent over $\mathbb{F}$.

Then there exists $c_0, c_1, \cdots, c_n \in \mathbb{F}$ with

$(c_0, c_1, \cdots, c_n) \neq (0, 0, \cdots, 0)$ such that

$c_0 * 1 + c_1 * \alpha + \cdots + c_n * \alpha^n = 0$

$\Rightarrow$ Let $f(x) = c_n * x^n + \cdots + c_1 * x + c_0 \in \mathbb{F}[x]$

Then $f(\alpha) = 0$.

Theorem 3.4.3
$\mathbb{F} \leq \mathbb{E}$: finite extension
$\mathbb{F} \leq \mathbb{K}$: finite extension
Then $\mathbb{F} \leq \mathbb{K}$: finite extension and $[\mathbb{K} : \mathbb{F}] = [\mathbb{K} : \mathbb{E}] * [\mathbb{E} : \mathbb{F}]$

Theorem 3.4.4
If $\mathbb{F} \leq \mathbb{E}$: field extension,
then $\overline{\mathbb{F}_{\mathbb{E}}} = \{\alpha \in \text{is algebraic over } \mathbb{F}\} \leq \mathbb{E}$

($\overline{\mathbb{F}_{\mathbb{E}}}$: $\mathbb{E}$ 안의 $\mathbb{F}$ 의 대수적 닫힘)

Definition 3.4.5
A field $\mathbb{F}$ is algebraically closed if every non-constant polynomial in $\mathbb{F}[x]$ has a zero in $\mathbb{F}$.

대수적으로 닫힌 체 $\mathbb{F}$는 $\mathbb{F}[x]$의 모든 비 상수 다항식들이 $\mathbb{F}$에 해를 가지는 성질을 만족하는 체 이다.

Theorem 3.4.6
A field F is algebraic closed
if and only if
every non-constant polynomial in F[x] factors.

Theorem 3.4.7
Every field $\mathbb{F}$ has an algebraic closure $\overline{\mathbb{F}}$.

위 정리는 Zorn's lemma, Axiom of choice, Well-ordering-principle 을 사용하여 증명한다.

제 4 장

Finite Field

Theorem 4.1
Let $\mathbb{F} \leq \mathbb{E}$: finite extension of degree n over a finite field $\mathbb{F}$.
If $\mathbb{F}$ has q elements, then $\mathbb{E}$ has q^n elements.

위 정리로 다음과 같은 따름정리를 얻을 수 있다.

Corollary 4.2
If $\mathbb{E}$ is finite field of char p, then $\mathbb{E}$ contains exactly p^n elements for some positive integer n.

다음은 유한체를 건설할 때 사용하는 중요한 정리이다.

Theorem 4.3
Let $\mathbb{E}$ be a field of ^{n} elements contained in an algebraic closure $\overline{\mathbb{Z}_p}$ of $\mathbb{Z}_p$.
The elements of $\mathbb{E}$ are precisely the zeros in $\overline{\mathbb{Z}_p}$ of the polynomial x^{p^n} - x in $\mathbb{Z}_p[x]$.

Theorem 4.4
If $\mathbb{E}$ is a field of prime char p with algebraic closure $\overline{\mathbb{E}}$, then x^{p^n} - x has p^n distinct zeros in $\overline{\mathbb{E}}$.

Theorem 4.5

Multiplicative group $< \mathbb{F}^*, * >$ of non zero elements of a finite field $\mathbb{F}$ is cyclic.

Corollary 4.6

A finite extension $\mathbb{E}$ of finite field $\mathbb{F}$ is a simple extension of $\mathbb{F}$.

Pf) Let α be a generator for the cyclic group $\mathbb{E}^*$, then $\mathbb{E} = \mathbb{F}(\alpha)$.

Theorem 4.7

체$\mathbb{F}$의 표수가 p이면 모든 $\alpha, \beta \in \mathbb{F}$ 에 대하여,
$(\alpha + \beta)^{p^n} = \alpha^{p^n} + \beta^{p^n}$ 이다.

Theorem 4.8

소수 p의 거듭제곱 p^n이 주어지면, 위수가 p^n인 유한체 GF(p^n)이 항상 존재한다.

Theorem 4.9

If $\mathbb{F}$ is any finite field, then for every positive integer n, there is an irreducible polynomial in $\mathbb{F}[x]$ of degree n.

위 정리를 끝으로 유한체 상에 정의된 기약다항식을 생성하는 법에 필요한 정리 소개를 마친다.

제 5 장

Conclusion

<결론1>

$\mathbb{Z}_p \subset \mathbb{F}$: finite field

$\rightarrow$ Deg : n 에 대해 $|\ \mathbb{F}\ | = p^n$

$\rightarrow$ $\mathbb{F}^*$: cyclic 하며 order가 p^n - 1 이다.

이곳의 모든 원소 $\alpha \in \mathbb{F}^*$ 는 $\alpha^{p^n - 1} = 1$ 을 만족한다.

$\therefore \alpha^{p^n} = \alpha$

<결론2>

Let $f(x) \in \mathbb{Z}_p[x]$ with deg $f(x) = n$ is irreducible over $\mathbb{Z}_p$

By Kronecker's theorem, there exist a field of order p^n such that $\mathbb{Z}_p[x]/< f(x) > = \mathbb{F}$

이 field는 $\mathbb{Z}_p(\alpha)$ 와 동형이다.(α 는 $f(x)$의 근이다)

$\rightarrow$ 결론 1에 의해 x^{p^n}는 α를 근으로 가지는 임의의 다항식이다.

$\rightarrow$ $f(x)$는 α를 근으로 가지는 최소다항식이다

$\therefore f(x) \mid x^{p^n}$ - x mod p

<결론3>

즉, 계수가 $\mathbb{Z}_p$에 있는 임의의 기약다항식 $g(x)$는 x^{p^n} - x 의 factor 중에 하나임을 알 수 있다.

또한 $g(x)$의 차수 d는 n의 약수이다.

제 6 장

Rabin Irreducibility Test

제 1 절 Rabin Irreducibility Pseudo Code

Algorithm 1 Rabin Irreducibility Test

Input: A monic polynomial f $\in F_q$[x] of degree n, and $p_1, \cdots, p_k$ all the distinct prime divisors of n.
Output: Either "f is irreducible" or "f is reducible".

for j := 1 to k **do**
  n_j := n/p_j;
end for
for i := 1 to k **do**
  g := gcd(f, $x^{q_i^n}$ - x mod f);
  if g $\neq 1$, **then**
    "f is reducible' and STOP";
end for;
g := x^{q^n} - x mod f;
if g $= 0$, **then**
  "f is irreducible"
else
  "f is reducible"
end if

부록 A

Rabin Irreducibility C Code

제 1 절 최대공약수 생성

```
void divide(int* matrix1, int* matrix2, int k, int p, int n)
{
        int tmp = 0;
        for (int i = 0; i < k - n; i++)
        {
                tmp = matrix1[i];

                for (int j = 0; j < n + 1; j++)
                {
                        matrix1[i + j] = (matrix1[i + j] - tmp * matrix2[j
                }
                /*
                for (int j = 0; j < k + 1; j++)
                {P
                        printf(" %d", matrix1[j]);
                }
                printf("\n");
                */
        }


}
```

```
void divide1(int* matrix1, int* matrix2, int k, int p, int n, int tmp1, int
{
        // printf("i am divide 1\n");
        int tmp = 0;
        for (int i = 0; i <= tmp1 - tmp2; i++)
        {
                tmp = matrix1[k + 1 - tmp1 + i];

                for (int j = 0; j < tmp2; j++)
                {
                        matrix1[k + 1 - tmp1 + i + j] = ((matrix2[n + 1 - t
                }
        }
        /*
        printf("GF :");
        for (int j = 0; j < k + 1; j++)
        {
                printf(" %d", matrix1[j]);
        }
        printf("\n");
        */

}

void divide2(int* matrix1, int* matrix2, int k, int p, int n, int tmp1, int
{
        // printf("i am divide 2\n");






        int tmp = 0;
```

```
	for (int i = 0; i <= tmp2 - tmp1; i++)
	{
		tmp = matrix1[n + 1 - tmp2 + i];

		for (int j = 0; j < tmp1; j++)
		{
			matrix1[n + 1 - tmp2 + i + j] = ((matrix1[n + 1 - t
		}

	}
	/*
	printf("f :");
	for (int j = 0; j < n + 1; j++)
	{
		printf(" %d", matrix1[j]);
	}
	printf("\n");
	*/
}
```

제 2 절 기약다항식 판정

```
void test()
{
	int check = 0;
	int p = 0; int n = 0; int a = 0; int tmp = 0; int b = 0; int c = 0

	printf("enter the prime number: ");
	scanf_s("%d", &p);
	printf("enter the degree: ");
	scanf_s("%d", &n);

	int k = 1;
```

```
for (int i = 0; i < n; i++)
{
        k *= p;
}

int* gf = (int*)calloc(sizeof(int), (k + 1));

int cnt = 0;

for (int i = 1; i < n; i++)
{
        if (n % i == 0)
        {
                gf[cnt] = i;
                cnt += 1;
        }
}

for (int i = 0; i < n; i++)
{
        printf("gf[%d] = %d\n", i, gf[i]);
}
printf("cnt = %d\n", cnt);

for (int i = 0; i < cnt; i++)
{
        tmp = 1;
        for (int j = 0; j < gf[i]; j++)
        {
                tmp *= p;
        }
        gf[i] = tmp;
}
```

```
for (int i = 0; i < n; i++)
{
        printf("gf[%d] = %d\n", i, gf[i]);
}



int* f = (int*)calloc(sizeof(int), (n + 1));
int* f1 = (int*)calloc(sizeof(int), (n + 1));
int* GF = (int*)calloc(sizeof(int), (k + 1));

printf("enter the polynomial which you want to test:\n");
for (int i = 0; i < n + 1; i++)
{
        scanf_s("%d", &f[i]);
}
printf("\n");




for (int i = 0; i < cnt; i++)
{
        for (int j = 0; j < n + 1; j++)
        {
                f1[j] = f[j];
        }





        for (int j = 0; j < k + 1; j++)
        {
                GF[j] = 0;
        }
```

```
GF[k - gf[i]] = 1;
GF[k - 1] = -1 % p;

while (1)
{
        int tmp1 = k + 1; int tmp2 = n + 1;

        for (int m = 0; m < k + 1; m++)
        {
                if (GF[m] == 0)
                {
                        tmp1 -= 1;
                }
                else
                {
                        break;
                }
        }

        for (int m = 0; m < n + 1; m++)
        {
                if (f1[m] == 0)
                {
                        tmp2 -= 1;
                }
                else
                {
                        break;
                }
        }
        printf("tmp1 = %d, tmp2 = %d\n", tmp1, tmp2);
```

```
for (int m = 0; m < k + 1; m++)
{
        GF[m] = (GF[m] + p) % p;
}
for (int m = 0; m < n + 1; m++)
{
        f1[m] = (f1[m] + p) % p;
}




if (tmp1 >= tmp2)
{
        divide1(GF, f1, k, p, n, tmp1, tmp2);
}
else
{
        divide2(f1, GF, k, p, n, tmp1, tmp2);
}


b = 0; c = 0;
for (int m = 0; m < k ; m++)
{
        b = b + ((GF[m] + p) % p);
}
for (int m = 0; m < n ; m++)
{
        c = c + ((f1[m] + p) % p);
}
```

```
        if ((b == 0 && GF[k] != 0) || (c == 0 && f1[n] != 
        {
                printf("continue!\n");
                break;
        }
        else if (b != 0 && c != 0)
        {
                continue;
        }
        else
        {
                printf("a it's reducible!\n");

                printf("GF :");
                for (int z = 0; z < k + 1; z++)
                {
                        printf(" %d", GF[z]);
                }
                printf("\n");

                printf("f :");
                for (int z = 0; z < n + 1; z++)
                {
                        printf(" %d", f1[z]);
                }
                printf("\n");




                check = 1;
                break;
        }
}
```

```
        if (check == 1)
        {
                break;
        }
}

/*****************************************************/
for (int i = 0; i < k + 1; i++)
{
        GF[i] = 0;
}

GF[0] = 1;
GF[k - 1] = -1 % p;



divide(GF, f, k, p, n);
/*
for (int i = 0; i < k + 1; i++)
{
        printf(" %d", GF[i]);
}
printf("\n");
*/
a = 0;

for (int j = 0; j < k + 1; j++)
{
        if (GF[j] == 0)
                a += 1;
}
```

```
/*
printf(" a = %d, k+1 = %d\n", a, k + 1);

for (int i = 0; i < n + 1; i++)
{
        printf("%dx^%d ", f[i], n - i);
}
*/

printf("\n");
if (n == 1)
{
        printf("it's irreducible\n");
}
else
{
        if (check != 1)
        {
                if (a == k + 1)
                {
                        printf("b it`s irreducible\n");
                }
                else
                {
                        printf("it`s reducible\n");
                }
        }
}


printf("\n");



free(gf);
```

```
        free(GF);
        free(f);
        free(f1);
        return 0;
}
```

제 3 절 기약다항식 생성

```
void make()
{
        int check1 = 0;
        int p = 0; int n = 0; int a = 0; int tmp = 0; int b = 0; int c = 0

        srand(time(NULL));

        printf("enter the prime number : ");
        scanf_s("%d", &p);
        printf("enter the degree : ");
        scanf_s("%d", &n);

        int k = 1;

        for (int i = 0; i < n; i++)
        {
                k *= p;
        }

        int* gf = (int*)calloc(sizeof(int), (k + 1));

        int cnt = 0;

        for (int i = 1; i < n; i++)
        {
```

```
        if (n % i == 0)
        {
                gf[cnt] = i;
                cnt += 1;
        }
}

for (int i = 0; i < n; i++)
{
        printf("gf[%d] = %d\n", i, gf[i]);
}
printf("cnt = %d\n", cnt);

for (int i = 0; i < cnt; i++)
{
        tmp = 1;
        for (int j = 0; j < gf[i]; j++)
        {
                tmp *= p;
        }
        gf[i] = tmp;
}

for (int i = 0; i < n; i++)
{
        printf("gf[%d] = %d\n", i, gf[i]);
}


int* f = (int*)calloc(sizeof(int), (n + 1));
int* f1 = (int*)calloc(sizeof(int), (n + 1));
int* GF = (int*)calloc(sizeof(int), (k + 1));
```

```
while (1)
{
        f[0] = 1;
        for (int i = 1; i < n + 1; i++)
        {
                f[i] = (rand() + p) % p;
        }

        for (int i = 0; i < n + 1; i++)
        {
                printf("%d ", f[i]);
        }
        printf("\n");

        for (int i = 0; i < cnt; i++)
        {
                for (int j = 0; j < n + 1; j++)
                {
                        f1[j] = f[j];
                }

                for (int j = 0; j < k + 1; j++)
                {
                        GF[j] = 0;
                }

                GF[k - gf[i]] = 1;
                GF[k - 1] = -1 % p;

                while (1)
                {
                        int tmp1 = k + 1; int tmp2 = n + 1;

                        for (int m = 0; m < k + 1; m++)
```

```
{
        if (GF[m] == 0)
        {
                tmp1 -= 1;
        }
        else
        {
                break;
        }
}

for (int m = 0; m < n + 1; m++)
{
        if (f1[m] == 0)
        {
                tmp2 -= 1;
        }
        else
        {
                break;
        }
}
printf("tmp1 = %d, tmp2 = %d\n", tmp1, tmp2


for (int m = 0; m < k + 1; m++)
{
        GF[m] = (GF[m] + p) % p;
}
for (int m = 0; m < n + 1; m++)
{
        f1[m] = (f1[m] + p) % p;
}
```

```
if (tmp1 >= tmp2)
{
        divide1(GF, f1, k, p, n, tmp1, tmp2
}
else
{
        divide2(f1, GF, k, p, n, tmp1, tmp2
}



b = 0; c = 0;
for (int m = 0; m < k ; m++)
{
        b = b + ((GF[m] + p) % p);
}
for (int m = 0; m < n ; m++)
{
        c = c + ((f1[m] + p) % p);
}

if ((b == 0 && GF[k] != 0) || (c == 0 && f1
{
        break;
}
else if (b != 0 && c != 0)
{
        continue;
}
else
{
```

```
                    check1 = 1;
                    break;
                }
            }

            if (check1 == 1)
            {
                break;
            }
        }

        /*****************************************************/
        for (int i = 0; i < k + 1; i++)
        {
            GF[i] = 0;
        }

        GF[0] = 1;
        GF[k - 1] = -1 % p;



        divide(GF, f, k, p, n);
        /*
        for (int i = 0; i < k + 1; i++)
        {
            printf(" %d", GF[i]);
        }
        printf("\n");
        */
        a = 0;

        for (int j = 0; j < k + 1; j++)
        {
            if (GF[j] == 0)
```

```
                    a += 1;
        }



        /*
        printf(" a = %d, k+1 = %d\n", a, k + 1);

        for (int i = 0; i < n + 1; i++)
        {
                printf("%dx^%d ", f[i], n - i);
        }
        */

        printf("%d\n",n);

        if (a == k + 1)
        {
                printf("it`s irreducible\n");
                break;
        }
        else
        {
                printf("it`s reducible\n");
        }
}

for (int i = 0; i < n + 1; i++)
{
        printf("%d ", f[i]);
}
printf("\n");

free(gf);
free(GF);
```

```
        free(f);
        free(f1);
        return 0;
}
```

제 4 절 rabin irreducibility code

4.1 main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
        int z = 0;

        while (1)
        {
                system("cls");
                printf("what do you want to do?\n1. test polynomial\n2. ma
                system("pause");

                scanf_s("%d", &z);

                system("cls");



                if (z == 1)
                {
                        test();
                        system("pause");
                }
                else if (z == 2)
                {
```

```
                make();
                system("pause");
            }
            else
            {
                printf("Thank you.\n");
                break;
            }
        }

        return 0;
}
```