여기 제시되는 논리들은 1 학기 [프로그래밍 논리의 이해]에서 다루고 연습한 내용입니다.

이를 숙지하고

몇 가지를 집중해서 풀어봅니다.

- isPrime 함수의 정의
- isSameArray 함수의 정의
- searchKey 함수의 정의
- reverse 함수의 정의

논리야 놀자 #/ – 반복문의 활용

프논에서 다룬 내용
논리는 아주 중요합니다.
반드시 다시 숙지하세요.

✪ **논리1: 1부터 n까지의 합을 계산(1 + 2 + 3 + 4 +…+ n)**

```
sum = 0;
i = 1;
while (i <= n) // n번 반복
{
    sum += i;
    i++;
}
```

✪ **논리2: 1부터 n까지의 곱을 계산(1 * 2 * 3 *…* n)**

```
product = 1;
i = 1;
while (i <= n) // n번 반복
{
    product *= i;
    i++;
}
```

✪ **논리3: 3을 n번 더함(3 + 3 + 3 + …+ 3)**

```
result = 0;
i = 1;
while (i <= n) // n번 반복
{
    result += 3;  result += 3;
    i++;
}
```

논리 3-2: m을 n번 더함(m + m + … + m)

```
result = 0;
i = 1;
while (i <= n) // n번 반복
{
    result += m;  result += 3;
    i++;
}
```

✪ **논리4: 5를 n번 곱함(5 * 5 * …* 5)**

```
result = 1;
i = 0; // 1 → 0
while (i < n) // n번 반복
{
    result *= 3;
    i++;
}
```

논리 4-2: m을 n번 곱함(m * m * … * m)

```
result = 1;
i = 0; // 1 → 0
while (i < n) // n번 반복
{
    result *= m;
    i++;
}
```

✪ **논리5: n개의 점수를 읽어서 총점을 계산, 평균을 계산(점수1 + 점수2 + …+ 점수n)**

```
total = 0;
i = 0;
while (i < n) // n번 반복
{
    score를 읽는다;
    total 누적;

    i++;
}
```

✪ **논리6: n개의 점수를 읽어서 최대값을 찾는다(점수1, …점수n중 가장 큰 값)**

```
max = 아주작은값;
i = 0;
while (i < n)
{
    score를 읽는다;
    if (max < score)
        max = score;

    i++;
}
```

논리야 놀자 #2

프논에서 다룬 내용

✪ 논리 7: n의 약수를 찾는다(찾아 출력한다)

약수의 성질은?

```
for (i = 1; i <= n; i++)
  if (n % i == 0)
        printf("%d\n", i);
```

✪ 논리 8: n이 소수인지 아닌지를 판별한다(이다 아니다를 출력)

소수의 성질은? 2부터 n-1까지 나눠지는 수가 있으면 소수가 아니다

아닌걸 발견하면 스톱,

소수라는 판단을 하려면 끝까지 봐야 안다.

```
prime = 1 //소수라고 일단 설정
i ← 2.. (n-1)
  if (n % i == 0)
     prime = 0; //소수 아님을 발견!!
     break;

if (prime)
  소수다
else
  소수 아니다
```

n이 소수인 경우는 1을 아니면 0을 반환하는 함수 isPrime(int n)을 정의하라.

아래와 같이 함수로 간단히 구현할 수있다!!

```
int isPrime(int n)
{




}
```

논리야 놀자 #3 – 배열(array)을 이용한 논리 연습

프논에서 다룬 내용

✪ 논리 9: 어떤 배열에 대해서 역순 배열 만들기

```
예: int A[5] = {10, 20, 30, 40, 50};
    int R[5];
    일 때 a의 원소들을 역순으로 r에 저장해보자.


A[] ← 10 20 30 40 50
R[] ← 50 40 30 20 10


i ← 0..4 // 즉 0..(크기 - 1)

   R[4 - i] ← A[i] // R[크기 - 1 - i] ← A[i]
```

✪ 논리10: 두 개의 배열이 같은가를 판별(하여 같다 다르다를 추력)

예: 두 개의 배열 A[], B[]를 비교해보자.

```
처음에는 일단 같다고 설정.
일단 크기가 다르면, 다르다
크기가 같다면
    첫번째, 두번째, ..., 마지막 원소를 비교하다가 다른 것이 발견되면, 다르다
    다름을 발견하면 스톱

명심할 점: 같다는 판단을 하려면 끝까지(두 배열의 마지막 원소까지 비교..) 해야 안다.

same ← 1;
If 크기가 다르면
    same ← 0;
else
    for (i ← 0.. 크기-1)
       다른 것이 발견되면 same ← 0하고 for문을 빠져나온다

If same
    같다
else
    다르다
```

☞같으면 1을, 다르면 0을 반환하는 isSameArray 함수에서는 return을 사용하므로 더 간단하게 코딩 가능하다. isSameArray함수를 작성해보라!

```
int isSameArray(int a[], int b[], int aSize, int bSize) // aSize, bSize는 각각 배열 a, b의 크기
{




}
```

✪ **논리11(정렬):** 배열의 값을 오름차순(혹은 내림차순)으로 정렬한다.
예: 선택정렬 - 배열 list[] = {5, 3, 8, 1, 2, 7}을 오름차순으로 정렬해보자.
최소값 원소를 첫째 원소랑 바꾸고, 그 다음의 최소값 원소를 둘째 원소랑 바꾸고…

(이 논리는 지난 학기에는 다루지 않았습니다. 다음 수업에서 다루어 봅시다)

프논에서 다룬 내용

✪ **논리12(탐색)**

프논에서 다룬 내용

배열에서 어떤 값(탐색키, search key)을 가진 원소가 있는가를 판별
(있다 없다, 혹은 몇 번째에 있다. 여러 개 있을 경우 첫 번째 것으로 한다)

예: array[12] = {11, 22, 33, 44, 55, 66, 11, 22, 33, 44, 55, 66}일 때

33이 배열에 있는가 없는가? 혹은 33이 배열의 몇 번째에 있는가?(배열안에 33이 여러개 있을 경우 첫번째것)

searchKey를 발견하면 스톱,
없는 것은 끝까지(배열의 마지막 원소까지)봐야 안다

다양한 방법으로 풀이가 가능합니다. 각자 풀어보고 다른 버전도 살펴봅시다.

SIZE를 배열의 크기라고 가정하자.

```
int a[SIZE] = {11, 22, 33, 44, 55, 66, 11, 22, 33, 44, 55, 66};
```

```
//버전1
searchKey를 읽는다 ;
```

```
//버전2
searchKey를 읽는다 ;
```

```
//버전3
searchKey를 읽는다 ;
```

함수로 정의해 봅시다. 함수 searchKeyInArray는 searchKey가 배열내에 있으면 그 index를, 없으면 -1을 반환합니다.

```
int searchKeyInArray(int a[], int size, int searchKey)
{

}
```

✪ **논리13**: 어떤 특정한 값을 갖는 원소들을 모아 배열에 넣는다.

예: 정수를 10개 읽으면서 홀수이면 배열 odd에 넣고 짝수이면 배열 even에 넣는다.

프논에서 다룬 내용

```
oddIndex ← 0;
evenIndex ← 0;
10번 반복하다
    수를 읽는다.
    그 수가 홀수이면,
        odd[oddIndex] ← 그 수
            oddIndex++;
    그렇지 않으면(짝수이면)
        even[evenIndex] ← 그 수
            evenIndex++;
```

논리야 놀자 #4 – 배열(array)을 이용한 논리 연습

프논에서 다룬 내용

✪ **논리 14: 두 수의 값을 바꾼다.**

a ← 5, b ← 10일 때
a ←→ b는 어떻게 하나?

```
int a = 5, b = 10;
int temp;
```

✪ **논리 15: 배열을 역순배열로 바꾼다**

만약 int a[5] = {1, 2, 3, 4, 5}일 때

a[0] ←→ a[4]
a[1] ←→ a[3]
a[2] ←→ a[2] - 할 필요 없음

만약 int a[6] = {1, 2, 3, 4, 5, 6}일 때

a[0] ←→ a[5]
a[1] ←→ a[4]
a[2] ←→ a[3]

```
for (i = 0; I < SIZE/2; i++)
    a[i] ←→ a[                    ];
```

// reverse 함수의 정의, 반환 타입? 매개변수? 지역변수?

프논10th 수업에서 다루어지는 것들: 배열(array)

프논에서 다룬 내용

3 개의 성적을 다루려 한다.

방법1: 3개의 변수를 사용한다.
int score1, score2, score3 라고 쓸 수 있다.

방법2: 배열을 사용한다. better!!!!!!!!

```
int scores[3];

// 3개의 정수형 변수를 가진 배열 scores 선언,
// scores[0], scores[1], scores[2]을 3개의 변수처럼 쓸 수 있다.
```

- **배열의 구성요소**

배열 원소의 형(type)/배열이름 /배열의 크기

배열 원소(element)

배열의 첨자(index)

- **배열의 접근 방법:** 이름과 첨자를 이용하여 마치 하나의 독립적 변수처럼 사용가능하다.

scores[0] : 첫번째 원소
scores[1] : 두번째 원소
scores[2] : 세번째 원소

- **배열 원소의 초기화**

1. 방법1: 원소별로 대입문 사용	2. 방법2:for문 안에서 대입문 사용	3. 방법3: 선언과 동시에 초기화

int score[3];
scores[0] = 10;
scores[1] = 10;
scores[2] = 10;
	int scores[3];
	for (i = 0; i < 3; i++)
	scores[i] = 10;
		int scores[3] = {10, 10, 10};

- **배열 원소의 사용 예: int scores[3]; // 선언되어있다고 가정**

//값배정, 10, 20, 30을 넣으려 할 때
for (i = 0; i < 3; i++)
scores[i] = i * 10;
	//입력
	
	for (i = 0; i < 3; i++)
	{
	printf("Enter a score:");
	scanf("%d", &scores[i]);
	}

//계산
total = 0;
for (i = 0; i < 3; i++)
total += scores[i];
total 출력;
	//출력
	for (i = 0; i < 3; i++)
	printf("%d ", scores[i]);

- **sizeof 연산자**
  sizeof(int)
  sizeof(배열의 이름) : 그 배열이 차지하는 메모리의 크기(byte단위로)

  예1: sizeof(scores) // 값은 12 = 3(배열의 크기) * 4(int의 크기)

  예2: *sizeof(scores) / sizeof(int) //*
  값은 배열의 크기, 즉 여기서는 3 = 12 / 4
- **기타: 배열의 크기는 미리 정해져야 함**
  그러므로 사용하려는 점수의 개수를 모를 때
  일단 넉넉하게 크기를 잡아놓고 사용한다.
  예: int scores[100];

```
printf("학생수(<=100)를 입력 "); //100보다 작은 수를 넣도록 입력프롬트에서 소개
scanf("%d", &num);
       // num이 100이하이므로 필요한 부분만 사용하고 나는 비어있음
       // 예를 들어 40명인 경우 0..39만 사용하고 40..99는 사용하지 않게된다.

for (i = 0; i < num; i++) //
{
...scores[i]...
}
```

LAB 1 : 1 차원 배열 연습

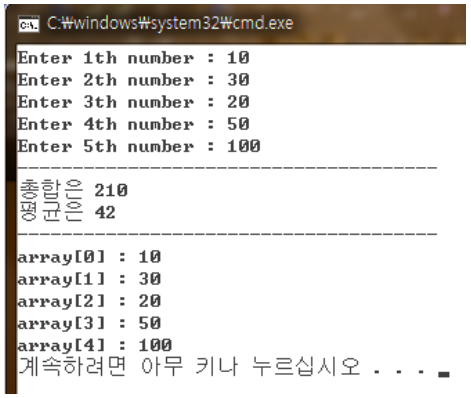
■ **LAB1_1(for 문, 배열) 배열의 내용 출력하기**

5 개의 정수를 배열로 입력 받아서 총합과 평균을 출력하고
그 배열의 내용을 화면에 출력하는 프로그램을 작성하라.

```
#include <stdio.h>
int main(void)
{
        int num[5];
        // 나머지 필요한 변수를 선언하라




        return 0;
}
```



■ **LAB1_2(for 문, 배열)난수 발생시켜서 출력해보기.**

rand()함수, srand()함수의 복습

1학기때 다루었던 함수로 srand()와 rand()가 있다.

rand()함수

rand() 함수는 random하게 0부터 RAND_MAX(32767)까지의 정수를 생성해준다.

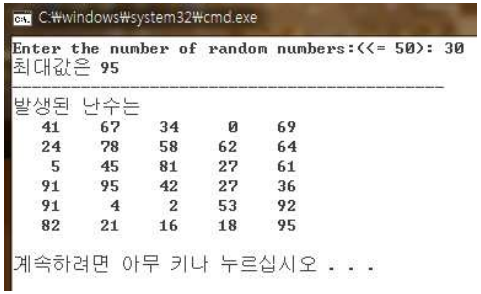
srand(*씨앗수*)

이 rand() 함수를 호출하기 전에 불러 주는 함수가 srand(*씨앗수*) 함수로써 *씨앗수*를 바꿀때마다 다른 임의의 수를 생성하게 된다.

어떤 수(50 이하의 수라고 가정)를 읽어서
1) 난수를 그 수만큼 발생시켜서 그 중 가장 큰 수를 출력하라.
2) 발생시킨 난수(0 부터 99 까지의)를 보기좋게 출력하고

- 난수를 발생시킬때 srand(time(NULL))을 일단 주석처리하였다. 난수 발생시 매번 다른 수가 나오도록 하는 것이 맞으나, 아래의 예는 srand(time(NULL))을 사용하지 않고 (의사)난수를 발생시켰다. 답이 맞는 가를 확인한 후 srand(time(NULL))을 활성화 시켜서 실행할때마다 다른 수가 나오도록 하라.

- 난수를 출력할 때 각 숫자는 5 개의 문자영역을 차지(%5d 사용)하고 오른쪽 줄맞춤 되도록 하라.




```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
      int data[100];
      int num, i;

      // srand(time(NULL));  // 주어진 실행 결과와 일치하나를 보기위해서 comment 처리함.
                             // 최종 실행시 comment 처리를 취소하고 실행시켜보세요.

      printf("Enter the number of random numbers:(<= 100): ");
      scanf("%d", &num);




      return 0;
}
```

HW 1 : 1 차원 배열 연습

■ **HW1_1**(for 문, 배열, rand()함수)(난이도 중?)

0 부터 9 까지의 출현 회수를 세어 출력하는 부분을 추가하라.

- srand(time(NULL))을 사용하지않고 아래의 결과와 일치하는지 확인한후(왼쪽 실행예), 숙제를 낼때는 srand(…)를 사용하여 제출하라.
- 0 부터 9 까지의 개수를 세기위해 크기가 10 인 정수형 배열 count 를 사용하라.
  즉, count[0]에 0 의 개수 저장, … , count[9]에 9 의 개수 저장

<srand 를 사용하지 않은 실행예>

```
C:₩WINDOWS₩system32₩cmd.exe
Enter the number of random numbers:(<= 100): 40
0의 개수는 1
1의 개수는 7
2의 개수는 6
3의 개수는 1
4의 개수는 5
5의 개수는 5
6의 개수는 3
7의 개수는 5
8의 개수는 4
9의 개수는 3
----------------------------------------
발생된 난수는
    1    7    4    0    9
    4    8    8    2    4
    5    5    1    7    1
    1    5    2    7    6
    1    4    2    3    2
    2    1    6    8    5
    7    6    1    8    9
    2    7    9    5    4

계속하려면 아무 키나 누르십시오 . . .
```

<실행예: 실행할때마다 다른 예가 나온다>

```
C:₩windows₩system32₩cmd.exe
Enter the number of random numbers:(<= 100): 10
0의 개수는 1
1의 개수는 0
2의 개수는 1
3의 개수는 0
4의 개수는 3
5의 개수는 1
6의 개수는 2
7의 개수는 0
8의 개수는 1
9의 개수는 1
----------------------------------------
발생된 난수는
    6    5    4    9    2
    8    4    0    6    4

계속하려면 아무 키나 누르십시오 . . .
```

■ **HW1_2**(난이도 중상)(응용) 자판기 프로그램 : 거스름돈 계산하기

주어진 거스름돈에 대해서 어떤 동전으로 거슬러 줄지를 계산하는 프로그램이다.
동전은 500 원, 100 원, 50 원, 10 원짜리 동전이 있으며, 거슬러 주는 동전의 개수를 최소한으로 하는 것을 목적으로 한다. 예를 들어 100 원을 거슬러 줄 경우, 10 원짜리 10 개를 거슬러 주는 것이 아니라, 100 원짜리 1 개를 거슬러 주어야 한다.

[주의사항]

1. 각 동전의 금액은 coins 라는 배열에 저장되어 있다.
2. **int coins[4] = {500, 100, 50, 10};**
3. 돌려주어야 할 금액이 change 이라는 변수에 저장되어 있다고 가정하면
4. 일단 금액이 큰 동전부터 시작한다.
5. 동전의 액면 금액(예를 들면, 500 원)으로 나누었을 때, 몫이 거슬러주어야 할 동전의 개수가 된다.
6. change 에 액면 금액으로 나눈 나머지를 넣고 다음 동전으로 넘어간다.