1. 포인터에 대한 설명을 읽고 설명이 맞으면 O, 틀리면 X를 선택하시오.

(1) 포인터는 다른 변수를 가리키는 변수이다. ( )

(2) 포인터의 크기는 포인터가 가리키는 변수의 크기와 같다. ( )

(3) 포인터는 다른 변수의 주소를 저장한다. ( )

(4) 포인터를 이용하면 이름을 직접 사용할 수 없는 변수에도 접근할 수 있다. ( )

(5) 포인터에 직접 절대 주소를 저장하고 사용할 수 있다. ( )

(6) int 포인터로 double 변수를 가리키고 사용해도 아무 문제없다. ( )

(7) 포인터를 초기화하지 않고 사용하는 것은 위험하다. ( )

(8) 포인터의 크기를 구할 때 sizeof 연산자를 이용한다. ( )

2. 널 포인터에 대한 설명 중 잘못된 것은?

① NULL은 <stdio.h>에 정의된 매크로 상수이다.

② NULL 매크로는 0으로 정의된 매크로이다.

③ 널 포인터에 대하여 역참조 연산자(*)를 사용하면 메모리 0번지에 값을 저장할 수 있다.

④ 포인터가 가리키는 변수가 없을 때 널 포인터로 만든다.

3. 다음 중 포인터의 선언이 잘못된 것을 모두 고르시오.

①
```
int arr[10];
int *p = arr;
```

②
```
double x;
double *pd = x;
```

③
```
int a;
short *ps = &a;
```

④
```
char *pc = NULL;
```

⑤
```
int z;
int *pz = &z;
```

4. 다음과 같이 포인터가 선언되어 있을 때 보기 중 잘못된 코드를 모두 고르시오.

```
double x;
double *pd = NULL;
```

①
```
pd = &x;
```

②
```
pd = &0.5;
```

③
```
pd = (x * 0.1);
```

④
```
pd = &x;
*pd = 0.5;
```

⑤
```
*x = 0.5;
```

5. 포인터 사용에 대한 설명 중 잘못된 것은?

① 주소 구하기 연산자(&)는 상수나 수식에는 사용할 수 없다.

② 포인터의 데이터형은 포인터가 가리키는 변수의 데이터형과 같아야 한다.

③ 포인터를 초기화하지 않으면 자동으로 0으로 초기화된다.

④ 역참조 연산자(*)는 포인터 변수에만 사용할 수 있다.

⑤ &를 사용하면 포인터의 주소도 구할 수 있다.

6. 다음과 같이 포인터가 선언되어 있을 때, 포인터 연산의 결과를 구하시오.

```
int data[] = { 1, 3, 5, 7, 9 };
int *p1 = data;     // p1에 저장된 data의 주소가 0x400번지라고 가정한다.
```

(1) *p1

(2) p1 + 1

(3) *(p1 + 1)

(4) p1[1]

(5) p1 + 3

7. 다음과 같이 포인터가 선언되어 있을 때, 포인터 연산의 결과를 구하시오.

```
int data[] = { 1, 3, 5, 7, 9 };
int *p2 = &data[2]; // p1에 저장된 data[2]의 주소가 0x408번지라고 가정한다.
```

(1) *p2

(2) p2[0]

(3) *(p2 + 1)

(4) p2[1]

(5) *(p2 + 2)

(6) p2[2]

8. 다음과 같이 const 포인터가 선언되어 있을 때, 컴파일 에러가 발생하는 코드를 모두 고르시오.

```
int data[] = { 1, 3, 5, 7, 9 };
int num[5];
const int *ptr = data;
```

① *ptr = 10;

② ptr = num;

③ ptr = &data[1];

④ (*ptr)++;

⑤ ++ptr;

⑥ ptr[1] = 30;

9. 다음 중 배열 이름이 포인터로써 가진 의미를 가장 잘 표현한 코드는?

```
int data[] = { 1, 3, 5, 7, 9 };
```

① int *data;
② const int *data;
③ int * const data;
④ const int * const data;

10. 다음과 같이 선언된 포인터에 대한 코드의 실행 결과 중 잘못된 것을 모두 고르시오.

```
short prices[] = { 1000, 2000, 3000, 4000 };     // prices의 주소는 0x200번지
short *ps = prices;                               // ps의 주소는 0x20A번지
```

코드	실행 결과
① printf("%d", *ps);	1000
② printf("%p", ps);	00000200
③ printf("%p", &ps);	0000020a
④ printf("%p", ps+1);	00000201
⑤ printf("%d", ps[3]);	4000
⑥ printf("%p", &ps[3]);	0000020c
⑦ printf("%d", sizeof(*ps));	2
⑧ printf("%d", sizeof(ps));	2

11. 배열과 포인터에 대한 설명을 읽고 설명이 맞으면 O, 틀리면 X를 선택하시오.

(1) 배열 이름은 배열의 시작 주소이다. ( )
(2) 배열 원소를 가리키는 포인터는 배열 이름인 것처럼 사용할 수 있다. ( )
(3) 포인터에 다른 배열의 시작 주소를 저장할 수 있다. ( )
(4) 배열 이름도 포인터처럼 증감 연산을 할 수 있다. ( )
(5) 배열의 시작 주소에 다른 배열의 시작 주소를 대입할 수 있다. ( )
(6) 배열의 시작 주소를 저장하는 포인터를 증가시키면 배열의 다음 원소를 가리키게 된다. ( )
(7) 배열의 시작 주소는 변경할 수 없다. ( )

12. 다음과 같이 선언된 배열과 포인터에 대해서 여러 가지 수식 중 의미가 다른 하나는?

```
short prices[10] = { 0 };
short *ps = prices;
```

① prices[i]

② *(prices + i)

③ *(ps + i)

④ ps + i

⑤ ps[i]

13. 다음과 같이 선언된 배열과 포인터에 대한 코드 중 컴파일 에러가 발생하는 것을 모두 고르시오.

```
double x[10], y[10], z[5];
double *pd = x;
```

① pd = y;

② pd = z;

③ x = y;

④ x = z;

⑤ ++pd;

⑥ pd += 3;

⑦ ++z;

14. 함수의 인자 전달 방법에 대한 설명 중에서 잘못된 것을 모두 고르시오.

① 값에 의한 전달 방법은 인자를 매개변수로 복사해서 전달한다.

② 함수 안에서 변경되는 인자는 포인터로 전달해야 한다.

③ void f(int arr[]);와 void f(int *arr);는 서로 다른 함수이다.

④ 배열을 함수의 인자로 전달할 때는 복사해서 전달한다.

⑤ 배열을 입력 매개변수로 전달하려면 const 포인터형의 매개변수를 사용한다.

⑥ 함수 안에서 매개변수로 전달받은 배열의 크기를 sizeof 연산자로 구할 수 있다.

15. 다음 프로그램의 실행 결과를 쓰시오.

```
#include <stdio.h>

int main(void)
{
    int arr[5] = { 1, 2, 3, 4, 5 };
    int *p = NULL;
    int i;
    p = arr;
    for (i = 0; i < 5; i++)
    {
        printf("%d ", ++(*p));
        p++;
    }
    printf("\n");
    return 0;
}
```

16. 다음 프로그램의 실행 결과를 쓰시오.

```
#include <stdio.h>

int main(void)
{
    int arr[5] = { 1, 2, 3, 4, 5 };
    int *p = NULL;
    int i;
    p = arr;
    for (i = 0; i < 5; i++)
    {
        printf("%d ",*p++);
    }
    printf("\n");
    return 0;
}
```

17. 두 정수의 최대 공약수와 최소 공배수를 구하는 함수의 원형을 작성하시오. 하나의 함수로 최대 공약수와 최소 공배수를 모두 구해야 한다.

18. 다음은 int 배열의 합계와 평균을 구하는 get_sum_average 함수의 정의이다. ___ 부분에 필요한 코드를 작성하시오.

```
#include <stdio.h>

int get_sum_average(_____________________________________________)
{
    int sum = 0;
    int i;
    for (i = 0; i < size; i++)
        sum += arr[i];
    if (average != NULL)
        *average = (double)sum / size;
    return sum;
}

int main(void)
{
    int scores[5] = { 98, 99, 78, 85, 91 };
    double ave;

    printf("합계 : %d\n", get_sum_average(scores, 5, &ave));
    printf("평균 : %.2f\n", ave);

    return 0;
}
```