Lab 7

두 명씩 짝을 지어 상의하면서 문제를 푼다.
각 문제는 명기한 시간 안에 완성하고 조교의 점검을 받고 다음 문제로 넘어갑니다.
각 문제별로 정해진 시간을 초과할 수 없습니다.

문제 1 (20분)

다음 프로그램을 이해하여 출력을 예측해본 뒤, 실제로 실행하여 예측이 맞았는지 확인해보자.

```
#include <stdio.h>

int main() {
    int i = 3, j = 7;
    int *p = &i, *q = &j;

    printf("*p + 7 = %d\n", *p + 7);
    printf("3 * **&p + 1 = %d\n", 3 * **&p + 1);
    printf("p = %p\n", p); /* 1 */
    printf("q = %p\n", q); /* 1 */
    printf("p - q = %d\n", p - q); /* 2 */
    printf("p - 2 = %p\n", p - 2); /* 3 */
    printf("p - (p - 2) = %d\n", p - (p - 2)); /* 4 */
}
```

1. 이 프로그램을 컴파일하면 컴파일러에 따라서 포인터 연산 때문에 주의 메시지(warning message)가 나올 수 있다. 주의 메시지가 나오면, 주의 메시지가 나오는 원인을 팀원과 함께 파악한 다음 프로그램을 수정하여 주의메시지가 나오지 않도록 하자. 주의 메시지가 나오지 않으면 다음 문제로 넘어간다.
2. `/* 1 */` 과 `/* 3 */` 문장이 프린트한 값들은 16진수로서 두 포인터 변수가 갖고 있는 주소를 표현한다. `%p`는 주소를 16진수로 프린트하는 포맷이다. 16진수가 친숙하지 않아서 읽기 힘들다. 포인터 변수의 주소값을 `unsigned long`으로 타입변환하여 십진수로 주소를 프린트하도록 프로그램을 수정하자.
3. `/* 1 */` 문장이 프린트한 주소값의 차이와 `/* 2 */` 문장이 프린트한 `p - q` 값을 비교해보자. 다르다. 왜 그럴까?
4. `/* 3 */` 문장과 `/* 4 */` 문장이 프린트한 값을 잘 관찰하여 3번에서 이해한 포인터 뺄셈의 원리를 확인하자.

문제 2 (10분)

다음 프로그램을 실행해보면, 사용하고 있는 컴퓨터에서 `char`, `int`, `double` 타입의 값을 저장하는 메모리 사이즈를 알 수 있다.

```
#include <stdio.h>

int main() {
    char *pc = NULL;
    int *pi = NULL;
    double *pd = NULL;

    printf("char = %lu byte\n", (unsigned long) (pc + 1));
    printf("int = %lu byte\n", (unsigned long) (pi + 1));
    printf("double = %lu byte\n", (unsigned long) (pd + 1));
}
```

이 프로그램의 실행 의미를 이해한 후, `short`, `long`, `float`, `long double` 타입의 값이 저장하는 메모리 사이즈도 알아볼 수 있도록 이 프로그램을 확장하자.

문제 3 (10분)

다음 프로그램을 실행한 결과를 관찰하고, 왜 그런 결과가 나왔는지 설명해보자.

```
#include <stdio.h>

int main() {
    char a[] = "xyz";
    char *p;
    int i;

    p = a;
    for (i = 0; i < 3; ++i)
        printf("%c\n", *p++);
    printf("a = %s\n", a);

    p = a;
    for (i = 0; i < 3; ++i)
        printf("%c\n", (*p)++);
    printf("a = %s\n", a);
}
```

문제 4 (10분)

다음 프로그램을 실행한 결과를 관찰하고, 왜 그런 결과가 나왔는지 설명해보자.

```
#include <stdio.h>

int main() {
    char *p = "xyz";
    char *q = "xyz";
    char r[] = "xyz";

    if (p == q)
        printf("p and q have the same address!\n");
    else
        printf("p and q have different addresses!\n");
    if (p == r)
        printf("p and r have the same address!\n");
    else
        printf("p and r have different addresses!\n");
}
```

사용하는 컴파일러에 따라서 결과가 다르게 나올 수 있음을 알아두자.

문제 5 (20분)

회문(palindrome)은 앞에서부터 읽으나 뒤에서부터 읽으나 똑같은 문자열을 말한다. 회문의 예를 들면 다음과 같다.

```
C
civic
kayak
madam
racecar
radar
rotator
step on no pets
no lemon no melon
```

문자열을 인수로 받아, 회문이면 `1`을 회문이 아니면 `0`을 내주는 함수를 다음 함수 프로토타입에 맞추어 작성하자.

```
int palindrome(char *, int);
```

첫째 인수는 문자열 포인터이고, 둘째 인수는 그 문자열의 길이이다.

위의 사례를 모두 테스트하는 `main` 함수도 추가로 작성한다.

문제 6 (20분)

초단위의 신간을 인수로 받아서 몇 시간 몇 분 몇 초인지 계산하여 결과를 포인터로 전달해주는 함수를 다음 함수 프로토타입에 맞추어 작성하자.

```
void time(int t, int *h, int *m, int *s);
```

예를 들어, 인수가 3723이면 1시간 2분 3초이므로 시, 분, 초 값을 각각 정수 포인터 인수로 전달해주어야 한다. 함수가 제대로 작동하는지 보여줄 수 있는 main 함수를 만들어 테스트한다.

문제 7 (30분)

2개의 정렬된 정수 배열을 하나의 정렬된 배열로 통합하는 함수를 다음 함수 프로토타입에 맞추어 작성하자.

```
void merge(int *x, int *y, int *z);
```

이 함수는 다음과 같은 절차로 배열 x와 y를 z로 복사한다.