```c
/* TelPhone1.c */

#include <stdio.h>
#include <string.h>

struct person {
	char name[20];
	char phone[20];
};

int main (void)
{
	struct person p;

	strcpy(p.name, "Free Lec");
	strcpy(p.phone, "02-3142-6702");

	printf("name : %s, phone : %s \n", p.name, p.phone);

	return 0;
}
```

```c
/* TelPhone2.c */

#include <stdio.h>

struct person {
	char name[20];
	char phone[20];
};

int main (void)
{
	struct person pArray[3];
	int i;

	for(i=0; i<3; i++)		//데이터 입력
	{
		printf("이름, 전화번호 순으로 입력 : ");
		scanf("%s %s", pArray[i].name, pArray[i].phone);
	}

	printf("\n입력 결과는 다음과 같습니다.\n");
	for(i=0; i<3; i++)		//데이터 출력.
	{
		printf("이름: %s, ", pArray[i].name);
		printf("전화번호: %s\n", pArray[i].phone);
	}

	return 0;
}
```

```c
/* struct_pointer1.c */

#include <stdio.h>

struct person {
	char name[20];
	char phone[20];
};

int main()
{
	struct person man={"Thomas", "354-00xx"};
	struct person * pMan;
	pMan=&man;

	// 구조체 변수를 이용한 출력.
	printf("name : %s\n", man.name);
	printf("phone : %s\n", man.phone);

	// 구조체 포인터를 이용한 출력.
	printf("name : %s\n", (*pMan).name);
	printf("phone : %s\n", (*pMan).phone);

	// 구조체 포인터를 이용한 출력.
	printf("name : %s\n", pMan->name);
	printf("phone : %s\n", pMan->phone);

	return 0;
}
```

```c
/* struct_pointer2.c */

#include <stdio.h>

struct perInfo {
	char addr[30];
	char tel[20];
};

struct person {
	char name[20];
	char pID[20];
	struct perInfo* info;
};

int main()
{
	struct perInfo info={"Korea Seoul", "333-4444"};
	struct person man={"Mr. Lee", "820204-xxxx512"};

	man.info=&info;

	printf("name : %s\n", man.name);
	printf("pID    : %s\n", man.pID);
	printf("addr : %s\n", man.info->addr);
	printf("tel    : %s\n", man.info->tel);

	return 0;
}
```

```c
/* struct_callby.c */
#include <stdio.h>

struct simple {
	int data1;
	int data2;
};

void show(struct simple ts);		// call-by-value
void swap(struct simple * ps);	// call-by-reference

int main()
{
	struct simple s={1, 2};

	show(s);		// s의 멤버 출력
	swap(&s);		// s의 멤버 data1, data2의 값 변경
	show(s);		// s의 변경된 멤버 출력

	return 0;
}

void show(struct simple ts)		// call-by-value
{
	printf("data1:%d, data2:%d\n", ts.data1, ts.data2);
}
void swap(struct simple * ps)	// call-by-reference
{
	int temp;
	temp=ps->data1;
	ps->data1=ps->data2;
	ps->data2=temp;
}
```

```c
/* overlapped.c */
#include <stdio.h>

struct point{
	int x;
	int y;
};

struct circle {
	struct point p;
	double radius;
};

int main()
{
	struct circle c1={10, 10, 1.5};
	struct circle c2={{30, 30}, 2.4};

	printf("[circle1] \n");
	printf("x:%d, y:%d \n", c1.p.x, c1.p.y);
	printf("radius:%f \n",c1.radius);

	printf("[circle2] \n");
	printf("x:%d, y:%d \n", c2.p.x, c2.p.y);
	printf("radius:%f \n",c2.radius);

	return 0;
}
```

```c
/* union.c */
#include <stdio.h>

union u_data{
	int d1;
	double d2;
	char d3;
};

int main (void)
{
	union u_data data;

	data.d2=3.3;
	printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

	data.d1=2;
	printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

	data.d3='a';
	printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

	return 0;
}
```