[3] 5분

1) \*size => &size

\*error => &error 왜냐하면, 포인터 변수의 입력값은 주솟값이어야 한다.

2) &b => b 왜냐하면, 배열의 이름은 주솟값임

3) 문제없다.

[4] 5분

1 2

0 2

1 2

foo1 = {n, &n}; 얕은 복사(초기화)

foo2 = foo1; 얕은 복사(변수 자체가 아니라, 변수가 가지고 있는 값이 foo2에 복사됨)

foo3 = &foo1; (foo1의 구조체 주솟값이 foo3에 복사됨, 따라서 foo3을 역참조하면 foo1 참조)

[5] 15분

<주의사항>

1/ 문자열의 끝에 ‘NULL’이 존재한다

2/ for문의 condition 조절을 잘 해야할 것

3/ for madam, strlen(str) = 5

주석처리 해놓은 것 보고 설명

[6] 10분

[7] 15분

왼쪽 답 = 2 3 1 3

(1), (2) static으로 선언된 s는, 함수가 끝나도 사라지지 않음, 첫 실행 시 1번만 초기화되고 값 유지

(3) = block에서 선언된 변수와 전역변수가 겹치면, 그 블록에서는 local variable을 따른다.

오른쪽 답 = 대문자 아님, 소문자임

Exam

I

S

F

Inal

X

1. arr[2] 그대로 문자열 출력
2. &arr[3][1] = s의 주솟값, 여기서 -1 => i의 주솟값을 역참조 => i출력
3. arr+4 = easy를 가리키는 block 주솟값, 역참조하면 easy의 주솟값, [2]하면? S 출력
4. (char\*\*)인 arr를, (char\*) arr로 캐스팅? => 1차원 배열로 생각 => arr[6]는? c,s,1,0,1,NULL,f니까 f출력!!
5. arr[1] + 1 = final의 주솟값 +1부터 출력 => final
6. \*(\*(arr+2)+1)) 인데, arr+2를 역참조하면 exam의 주솟값, +1하면, x의 주솟값, 이를 역참조 => x출력

[9]

Void 포인터의 경우, 개념적인 부분은 좀 확실히 알고 있으면 외울 필요는 없다.

<유효한 포인터 수식>

포인터 자체는 ‘주소’의 개념이므로 타입에 구애받지 않는다.

<잘못된 포인터 수식>

하지만 그것을 역참조하려면, 타입마다 차지하는 size가 다르므로, type이 정해져있어야 한다.

P = PV 이것은, 넣을 수는 있지만 PV가 나중에 어떤 type으로 캐스팅 될지, 또 어떤 type의 변수에 대한 주솟값을 저장하고 있는지 모르기에 꺼림직한 표현

[11]

Calloc과 malloc의 차이, calloc은 할당과 동시에 메모리를 “0”으로 초기화한다!!

[12] 실습 (10분)

설명하고 끝