**Test #6. 2012.4.16 프로그래밍언어론 00 분반- 개인 문제**

학번:

이름:

조번호:

역할 : 사회자, 일반 조원 ........................ (자신의 팀 내의 역할에 동그라미 하시오.)

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. C에서는 포인터간의 뺄셈이 가능하다.
3. 일반 포인터를 void \* 타입의 포인터로 지정할 때는 (void\*)와 같은 casting 이 필요하다.
4. s가 구조체를 가리키는 포인터이고 f가 필드이름일 때, (\*s).f 는 s->f와 동일한 표현이다.
5. Garbage란 사용자 프로그램에서 더 이상 접근될 수 없는, 할당된 heap-dynamic 변수를 말한다.
6. 실제 포인터가 직접 heap-dynamic 변수를 가리키지 않고 대신 비석(tombstone)을 가리키도록 구현하면, 나중에 heap-dynamic 변수가 반환될 때 포인터는 nil을 가리키게 된다.
7. Garbage collection 에서는 자신을 참조하는 포인터의 개수가 0일 때 제거된다.
8. ~~Reference를 사용하면 pointer에서 발생되는 dangling reference가 항상 방지된다.~~

(reference는 dangling reference가 없다는 것으로 교재에 맞게 정정함.)

1. 포인터 p1이 새롭게 생성된 heap-dynamic 변수를 가리키도록 설정하고, 나중에 p1이 다른 heap-dynamic 변수를 가리키도록 설정하면, lost heap dynamic 변수가 생성된다.
2. C#은 C의 포인터를 제공하나 포인터 사용시 unsafe 사용이 요구된다.
3. Java에는 명시적 회수 기능이 없어서 dangling reference가 발생하지 않는다.

2. dangling reference (허상 참조)가 발생되었을 때의 문제점을 간단히 적으시오.

3. Pascal의 union의 안전성과 효율성에 대해 간단히 적으시오.

**Test #6. 2012.4.16 프로그래밍언어론 00 분반 조별 문제**

조번호:

참여자 (학번, 이름)

-

-

-

-

-

-

1. Garbage collector 가 작동하는 경우라도 프로그래머의 부주의나 알고리즘적인 문제로 적지 않은 memory leak (메모리 누수)가 발생할 수 있다. 이러한 예를 1~2 가지 생각해서 적어보시오.