**Test #6. 2013.4.16 프로그래밍언어론 02 분반**

학번:

이름:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. 참조 타입 변수에 대해서는 산술연산이 불가능하다.
3. 다음과 같은 C 변수 a와 b 가 있다고 가정하자.

struct A {

int f;

} a;

int b[1];

만일 int i=0;와 같은 선언이 추가되었을 때,

a.f 보다 b[i]를 접근하는 속도가 더 빠르게 된다.

1. 다음과 같은 코드에서 마지막 줄의 strlen 함수 호출 시 C는 컴파일 오류를 낸다.

union number {

int value;

char data[4];

} x;

x.value = 100; ...

strlen(x.data);

1. void\* 타입 변수는 항상 char\* 타입 변수와 크기가 같다.
2. 다음 코드는 분실된(lost) heap-dynamic 변수 문제를 안고 있다.

char c;

char \* p1= &c;

p1 = (char\*) malloc(sizeof(char));

1. 비석 (tombstone) 방법은 ( ) 문제를 해결하고자 나온 방법이다. ( )에 알맞은 말은?
2. Java에 dangling reference가 없는 이유를 간단히 설명하시오.

답

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. 참조 타입 변수에 대해서는 산술연산이 불가능하다. O
3. 다음과 같은 C 변수 a와 b 가 있다고 가정하자. X

struct A {

int f;

} a;

int b[1];

만일 int i=0;와 같은 선언이 추가되었을 때,

a.f 보다 b[i]를 접근하는 속도가 더 빠르게 된다.

1. 다음과 같은 코드에서 마지막 줄의 strcpy 함수 호출 시 C는 컴파일 오류를 낸다. X

union number {

int value;

char data[4];

} x;

x.value = 100; ...

strlen(x.data);

1. void\* 타입 변수는 항상 char\* 타입 변수와 크기가 같다. O
2. 다음 코드는 분실된(lost) heap-dynamic 변수 문제를 안고 있다. X

char c;

char \* p1= &c;

p1 = (char\*) malloc(sizeof(char));

1. 비석 (tombstone) 방법은 ( dangling reference ) 문제를 해결하고자 나온 방법이다. ( )에 알맞은 말은?
2. Java에 dangling reference가 없는 이유를 간단히 설명하시오.

Garbage collection에 의해 묵시적으로 불필요한 heap 데이터를 제거하며, 자신을 가르키는 것이 없는 셀만 제거 (dispose) 하므로, 다른 메모리에서 가리키고 있는 것은 절대로 제거하지 않는다. 따라서 reference 되고 있는 것은 절대 제거되지 않으므로, 개발자 실수로 도입되는 dangling reference를 막을 수 있다.