프로그래밍언어론

문 1. C, C++, Java 등의 언어에서 사용되는 동적 기억장소 할당에 대한 설명 중 옳은 것은?

① 프로그램 수행을 시작할 때 프로그램 수행에 필요한 전체 메모리의 크기가 결정된다.
② 쓰레기 수집(garbage collection)을 지원하는 언어의 경우 허상 포인터(dangling pointer) 문제가 발생하지 않는다.
③ 동적으로 할당되는 기억장소는 스택 영역의 활성화 레코드(activation record)에 배정된다.
④ C 언어에서는 쓰레기 수집(garbage collection) 방법을 사용하여 메모리를 관리한다.

문 2. 객체지향언어의 상속성을 구현하기 위한 프로그래밍 언어별 특징 및 적용에 대한 설명 중 옳지 않은 것은?

① Java 언어는 클래스의 다중상속을 허용하며, 예약어로 implements를 사용한다.
② Java 언어는 상속을 허용하며, 예약어로 extends를 사용한다.
③ C++ 언어는 다중상속을 허용하며, 상속하는 상위클래스들을 콤마(,)로 연결한다.
④ C++ 언어는 상속을 허용하며, 하위클래스는 상속하는 상위클래스를 콜론(:)으로 연결한다.

문 3. 다음 Java 프로그램에 사용된 객체지향언어의 특성이 아닌 것은?

```
public class Animal {
    private int legs = 4;
    String name = "동물";
    public void walk() {
      System.out.println(name + "(이)가 걸었습니다.");
    }
}
public class Lion extends Animal {
    String name = "사자";
    public void walk() {
      System.out.println(name + "가 걸었습니다.");
    }
}
```

① 캡슐화(encapsulation)
② 오버로딩(overloading)
③ 상속(inheritance)
④ 오버라이딩(overriding)

문 4. 웹브라우저에서 실행되는 자바 프로그램인 애플릿(applet)은 main()함수로부터 수행을 시작하는 C 프로그램과는 다른 실행 순서를 가지고 있다. 애플릿의 실행 순서로 옳은 것은?

① init() → service() → stop()
② start() → run() → stop()
③ start() → run() → paint() → destroy()
④ init() → start() → stop() → destroy()

문 5. 다음 Java 프로그램의 출력 결과는?

```
class SuperClass {
    int a, b;
    SuperClass() {
        a=1;
        b=1;
    }
    SuperClass(int a, int b) {
        this.a = a;
        this.b = b;
    }
}
class SubClass extends SuperClass {
    int c;
    SubClass() {
        c=1;
    }
    SubClass(int a, int b, int c) {
        super(a, b);
        this.c = c;
    }
}
public class Test {
    public static void main(String[] args) {
        SubClass o1 = new SubClass();
        SubClass o2 = new SubClass(1, 2, 3);
        System.out.println("a="+o1.a+", b="+o1.b+", c="+o1.c);
        System.out.println("a="+o2.a+", b="+o2.b+", c="+o2.c);
    }
}
```

① a=1, b=1, c=1
   a=1, b=1, c=1
② a=1, b=1, c=1
   a=1, b=2, c=3
③ a=1, b=2, c=3
   a=1, b=1, c=1
④ a=1, b=2, c=3
   a=1, b=2, c=3

문 6. 다음 문법으로 생성되지 않는 문장은?

```
<S> ::= 1<A> | 0<B>
<A> ::= 0<A> | 1<B>
<B> ::= 1<B> | 0
```

① 0110
② 1110
③ 01100
④ 100110

문 7. 다음 조건에 대해서 B가 2차원 배열이고, 행우선(row-major) 순서로 저장되어 있을 때 원소 B(i, j)의 주소를 계산하는 식은?

> ○ B(i, j) : 2차원 배열 B의 i행 j열 원소
> ○ E : 한 원소의 크기
> ○ LB1, LB2 : 행의 하한 값, 열의 하한 값
> ○ UB2 : 열의 상한 값
> ○ S : 한 행의 크기로 UB2 - LB2 + 1
> ○ base : 주 기억 장소에서 배열의 시작 주소

① base + [(j + LB1)S + (i + LB2)]E
② base + [(i + LB2)S + (j + LB1)]E
③ base + [(j - LB2)S + (i - LB1)]E
④ base + [(i - LB1)S + (j - LB2)]E

문 8. 다음 C 프로그램 코드가 실행된 후, 변수 n의 값으로 옳은 것은?

```
int i, j, n = 0;
for (i = 0; i < 10; i++) {
    for (j = 0; j < 10; j++) {
      if (j > 5) continue;
      n++;
    }
    if (i > 6) break;
}
```

① 30
② 36
③ 42
④ 48

문 9. 수식 1 + 4 * 0.5 + 2.5 + 3의 값을 계산할 때 일반적인 수학 원칙과는 다르게 아래에 주어진 원칙이 사용된다면 수식의 계산 결과는?

<계산 원칙>

> ○ +가 *보다 더 높은 우선순위(precedence)를 갖는다.
> ○ +와 *는 각각 덧셈과 곱셈 연산자로서 결합 순서(associativity)는 우측 우선(right associative)이다.
> ○ +와 *의 피연산자 자료형이 실수와 정수로 서로 다르면, 실수를 정수로 바꾸는 축소 변환(narrowing conversion)이 실수 피연산자에 대하여 자동으로 적용된다.
> ○ 실수에서 정수로의 자료형 변환에는 버림이 사용된다.

① 8.5
② 21
③ 25
④ 30

문 10. 다음 C 프로그램에 대한 설명 중 옳지 않은 것은?

```
(1) int *x, *y, *z;
(2) x = (int *) malloc(sizeof(int));
(3) *x = 1;
(4) z = (int *) malloc(sizeof(int));
(5) *z = 5;
(6) y = x;
(7) *y = 2;
(8) z = y;
(9) free(x);
```

① 문장 (6)이 수행된 이후에 *x와 *y의 별명(alias) 현상이 발생한다.
② 문장 (8)이 수행된 이후에 *x와 *y의 별명(alias) 현상은 종료된다.
③ 문장 (8)이 수행된 이후에 문장 (4)에서 할당된 기억장소는 쓰레기(garbage)가 된다.
④ 문장 (9)가 수행된 이후에 허상 포인터(dangling pointer)가 발생한다.

문 11. 어떤 변수에 대한 타입 바인딩(type binding)은 컴파일 시에 정적 타입 바인딩(static type binding)을 할 수도 있고 수행 시에 동적 타입 바인딩(dynamic type binding)을 할 수도 있다. 이 두 가지 방법에 대한 설명 중 옳지 않은 것은?

① 정적 타입 바인딩을 하는 이유는 수행 시간을 빠르게 하기 위함이다.
② 동적 타입 바인딩을 하는 이유는 융통성(flexible) 있게 데이터를 처리하기 위함이다.
③ 정적 타입 바인딩을 하는 프로그래밍 언어는 재귀 호출(recursion)을 수행할 수 없다.
④ 동적 타입 바인딩을 하는 언어는 대부분 인터프리터 방식으로 구현된다.

문 12. 객체지향 프로그래밍에 대한 설명 중 옳지 않은 것은?

① 객체지향 프로그래밍의 특징은 추상 데이터 타입(abstract data type)에 상속과 동적 바인딩(dynamic binding)이 추가된 것이다.
② 객체지향 프로그래밍 언어는 상속(inheritance)이란 특성을 통해 소프트웨어의 재사용성(reusability)을 향상시켰다.
③ 객체지향 프로그래밍을 실제로 프로그램 작성에 적용할 때 기본 단위는 함수이며, 이러한 함수들이 모여 프로그램을 구성하게 된다.
④ 상속은 상속 관계로 맺어진 클래스들 사이에 종속성을 만들므로, 추상 데이터 타입의 가장 큰 장점인 독립성을 훼손하는 측면이 있다.

문 13. 고급 언어(high-level language)에서 변수를 선언하는 목적에 해당하지 않는 것은?

① 변수의 크기를 지정
② 변수의 영역(scope)을 지정
③ 변수의 자료형을 지정
④ 변수의 위치를 지정

문 14. Java 언어의 특징과 관련 효과로서 가장 연관성이 적은 것은?

① 애플릿(applet) 프로그래밍 지원 – 웹(web)을 위한 언어로 널리 사용
② 유니코드 지원 – 메모리의 효율적 사용
③ 배열 첨자(index)의 범위 검사 – 배열에 대한 경계 이탈(out of bound) 오류 방지
④ 가상 기계(virtual machine) 기반 – C 언어보다 이식성이 우수

문 15. 기억장소 바인딩(binding)에 대한 설명 중 옳은 것은?

① Java 언어에서 static 예약어를 붙여서 선언한 변수는 스택-동적 변수(stack-dynamic variable)이다.
② 정적 변수(static variable)는 기억 장소가 실행시간 스택(run-time stack)에 할당된다.
③ 기억장소 회수(storage deallocation)는 프로그램이 주기억장치로부터 보조기억장치로 옮겨지는 것을 말한다.
④ 기억장소 할당(storage allocation)은 프로그램이 실행되기 위해서 필요한 변수들의 기억장소를 배정하는 것이다.

문 16. 다음 C 프로그램 코드에서 정수형 변수 a가 저장되어 있는 메모리 주소는 1000번지, 문자형 변수 b가 저장되어 있는 주소는 2000번지라고 가정하였을 때, 프로그램 코드가 실행된 후 i_ptr과 c_ptr에 저장되어 있는 값은?
(단, 정수형 변수는 4바이트가 할당된다고 가정한다)

```
int a, *i_ptr;  /* a 변수의 저장 주소는 1000번지 */
char b, *c_ptr; /* b 변수의 저장 주소는 2000번지 */
i_ptr = &a;
c_ptr = &b;
i_ptr = i_ptr + 2;
c_ptr = c_ptr + 2;
```

① i_ptr : 1002, c_ptr : 2002
② i_ptr : 1008, c_ptr : 2008
③ i_ptr : 1004, c_ptr : 2004
④ i_ptr : 1008, c_ptr : 2002

문 17. 다음은 프로세스 P1, P2, P3과 main을 실행하는 주 프로세스로 구성된 총 네 개의 프로세스가 병행적(concurrently)으로 실행되는 프로그램이다. 네 개의 프로세스가 공유하는 전역 변수 x의 초기값은 0이고, P1, P2, P3은 공유되지 않는 자신의 지역변수 y를 갖고 있다. P1, P2, P3은 주 프로세스에 의하여 동시에 실행이 시작된다. 선언문 혹은 할당문(assignment statement)은 모두 원자적(atomic)으로 실행된다고 가정한다.

```
global variable int x = 0;
process P1
    local variable int y = 0;
    y := x;
    x := y + 1;
endprocess P1
process P2
    local variable int y = 0;
    y := x;
    x := y + 2;
endprocess P2
process P3
    local variable int y = 0;
    y := x;
    x := y + 4;
endprocess P3
main
    start {P1, P2, P3}
endmain
```

세 개의 프로세스가 모두 종료된 후에 공유변수 x가 가질 수 없는 값은?

① 0
② 4
③ 6
④ 7

문 18. 다음 문맥무관문법(context-free grammar)에 대한 설명 중 옳지 않은 것은?

```
E -> E ⓐ T
   | T
T -> T ⓑ F
   | F
F -> id
   | ( E )
```

① 연산자 ⓐ는 연산자 ⓑ보다 우선순위가 높다.
② 괄호 연산자 '('와 ')'는 연산자 ⓐ보다 우선순위가 높다.
③ 연산자 ⓑ의 결합 순서(associativity)는 좌측 우선(left associative)이다.
④ 위의 문법은 모호하지 않다.

문 19. 다음 C++ 프로그램의 출력 결과는?

```
#include<iostream.h>
void inc_counter1(int *counter)
{
    ++(*counter);
}
void inc_counter2(int &counter)
{
    ++counter;
}
void inc_counter3(int counter)
{
  ++counter;
}
void main()
{
  int a_count = 0;
  inc_counter1(&a_count);
  inc_counter2(a_count);
  inc_counter3(a_count);
  cout<<a_count<<'\n';
}
```

① 0　② 1
③ 2　④ 3

문 20. 다음 C 프로그램의 출력 결과는?

```
#include<stdio.h>
void func(void)
{
    static int a = 200;
    int b = 1;
    printf("a=%d, b=%d \n", ++a, ++b);
    printf("a=%d, b=%d \n", a++, b++);
}
void main()
{
    int i;
    for(i = 1; i <= 3; i++) func();
}
```

① a=201, b=2
a=201, b=2
a=203, b=2
a=203, b=2
a=205, b=2
a=205, b=2

② a=201, b=2
a=201, b=2
a=201, b=2
a=201, b=2
a=201, b=2
a=201, b=2

③ a=201, b=2
a=202, b=3
a=203, b=2
a=204, b=3
a=205, b=2
a=206, b=3

④ a=200, b=1
a=202, b=3
a=202, b=1
a=204, b=3
a=204, b=1
a=206, b=3