프로그래밍언어론

문 1. C 언어에서 구조체와 공용체에 대한 설명으로 옳은 것은?
① 배열은 구조체의 멤버가 될 수 없다.
② 구조체를 구성하는 멤버는 구조체가 될 수 없다.
③ 같은 기억 장소를 여러 개의 변수가 공유하도록 하기 위해 공용체를 사용한다.
④ 구조체의 포인터를 통해 구조체의 멤버를 직접 참조할 때에는 '점(.)' 연산자를 사용한다.

문 2. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

typedef struct p {
    int x, y;
} PT;
void calcPoint(PT x, PT y, PT z) {
    z.x = x.x + y.x;
    z.y = x.y + y.y;
}
int main() {
    PT p1 = {10,20}, p2 = {30,40}, p3 = {0,0};
    calcPoint(p1, p2, p3);
    printf("(%d, %d)\n", p3.x, p3.y);
    return 0;
}
```

① (0, 0) ② (10, 20)
③ (30, 40) ④ (40, 60)

문 3. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

int main() {
    int a;
    double c, d;
    c = a = 3.5;
    d = 3+a/2+c;
    printf("Result = %3.1f\n", d);
    return 0;
}
```

① Result = 6.5 ② Result = 7.0
③ Result = 7.5 ④ Result = 8.0

문 4. 프로그래밍 언어에서 변수의 *l*-값(*l*-value)과 *r*-값(*r*-value)에 대한 설명으로 옳지 않은 것은?
① C 언어에서 변수 i의 주소를 저장한 포인터 변수 p의 경우, p의 *l*-값은 p 값이 저장된 위치를, *r*-값은 i의 값을 의미한다.
② 일반적인 프로그래밍 언어에서 "A := B"와 같은 배정문(assignment statement)이 있을 때, A의 *l*-값과 B의 *r*-값을 사용하게 된다.
③ Bliss 언어에서 모든 변수 이름은 언제나 *l*-값을 의미하므로 *r*-값을 표현하려면 별도의 단항 연산자를 사용해야 한다.
④ "a+b"와 같은 수식의 경우에는 *r*-값만 의미가 있으며 *l*-값은 사용하지 않는다.

문 5. 객체지향 언어의 추상클래스(abstract class)에 대한 설명으로 옳지 않은 것은?
① 여러 서브클래스에서 구현해야 할 동일한 인터페이스의 메소드를 선언하기 위해 사용한다.
② 추상클래스의 서브클래스를 정의하지 않아도 추상클래스의 인스턴스(instance)를 생성할 수 있다.
③ 서브타입 다형성(subtype polymorphism)을 제공하기 위해 사용한다.
④ 추상클래스의 서브클래스가 추상클래스인 경우에는 추상 메소드를 오버라이딩(overriding)하지 않아도 된다.

문 6. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i, res = 0;
    int *ip = (int *) calloc(5, sizeof(int));
    for (i = 0; i< 5; i++)
        ip[i] += i + 1;
    ip = (int *) realloc(ip, 10 * sizeof(int));
    for (i = 5; i< 10; i++)
        ip[i] = ip[i-5] + i + 1;
    for (i = 0; i< 10; i++)
        res += ip[i];
    printf("%d\n", res);
    return 0;
}
```

① 15 ② 40
③ 55 ④ 70

문 7. 다음 C 프로그램의 실행 결과로 15가 출력되기 위해 ㉠에 들어갈 값으로 옳은 것은?

```
#include <stdio.h>

#define COL   ( ㉠ )

int main() {
    int arr[ ][COL] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    const int ROW = sizeof(arr) / sizeof(int) / COL;
    int (*ptr1)[COL] = arr;
    int *ptr2[ROW];
    int i, res = 0;
    for (i = 0; i< ROW; i++) {
        ptr2[i] = *(ptr1 + i);
        res += *ptr2[i];
    }
    printf("%d\n", res);
    return 0;
}
```

① 2 ② 3
③ 4 ④ 5

문 8. 타입(type)에 대한 설명으로 옳지 않은 것은?

① 타입은 값들의 집합과 그 값들에 대한 연산의 집합이다.

② 타입 오류는 어떤 연산을 해당하지 않는 타입의 값에 적용함으로써 발생하는 오류를 의미한다.

③ 동적으로 타입이 결정된다(dynamically typed)는 것은 변수의 타입이 저장되는 값에 따라 실행 중에 바뀔 수 있는 것을 의미하며 Python과 같은 언어가 이에 속한다.

④ 엄격한 타입(strongly typed) 언어는 프로그래밍 언어의 타입 체계가 컴파일 중이나 실행 중에 모든 타입 오류를 찾아낼 수 있으며 정적으로 타입이 결정된다(statically typed)는 것을 의미한다.

문 9. 다음 Java 프로그램의 Test.main()의 실행 결과로 옳은 것은? (단, Java에서 static은 정적 필드나 정적 메소드를 정의하기 위한 구문이다)

```
class A {
    protected static int a=0;
    protected int b=0;
    public static void inc_a() { a++; }
    public void inc_b() { b++; }
    public void print() {
        System.out.println(a+" "+b);
    }
}

class B extends A {
    public static void inc_a() { a+=2; }
    public void inc_b() { b+=2; }
}

public class Test {
    public static void main(String[ ] args) {
        B y = new B();
        A z = new B();
        y.inc_a();
        y.inc_b();
        z.inc_a();
        z.inc_b();
        z.print();
    }
}
```

① 2 4　　② 4 2

③ 3 2　　④ 4 4

문 10. Java 언어에서 배열에 대한 설명으로 옳은 것은?

① 배열 선언 후 배열의 메모리를 할당하려면 new 연산자를 사용해야 한다.

② 배열 원소를 사용하기 전에 배열의 메모리를 할당할 필요는 없다.

③ 배열 원소에는 참조 값(reference value)을 저장할 수 없다.

④ 배열을 선언할 때에는 배열 크기를 지정해야 한다.

문 11. C 언어의 포인터 변수에 대한 설명으로 옳지 않은 것은?

① 첨자 없이 표현된 배열 이름에 값을 배정할 수 있다.

② 포인터 변수에 대한 제한된 형태의 산술 연산이 가능하다.

③ 포인터 p, 정수 index에 대해 “p + index”는 index의 값에 먼저 메모리 셀의 크기를 곱한 후 그 결과를 p에 더한다.

④ 배열 이름은 배열의 첫 번째 원소의 주소를 가리킨다.

문 12. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

int main() {
    int a, b;
    a = 8;
    b = 12;
    printf("a^b = %d\n", a^b);
    return 0;
}
```

① a^b = 1

② a^b = 2

③ a^b = 3

④ a^b = 4

문 13. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

#define MULTIPLY(a, b) (a*b)

int main() {
    float f = 100/MULTIPLY(2+3, 4+5);
    printf("%.1f\n", f);
    return 0;
}
```

① 2.0

② 5.0

③ 67.0

④ 180.0

문 14. 객체지향 언어에서 상속(inheritance)에 대한 설명으로 옳지 않은 것은?

① 다중 상속(multiple inheritance)의 상속 그래프를 그리면 방향성 비순환 그래프(directed acyclic graph)가 되므로 메소드가 둘 이상의 경로를 통하여 상속될 수 있다.

② Java에서는 클래스의 다중 상속을 허용하지 않는 대신에 인터페이스(interface)를 이용하여 다중 상속을 사용할 수 있다.

③ C++에서 다중 상속의 문제점을 해결하기 위하여 프렌드 함수(friend function)를 도입하였다.

④ Java에서는 객체가 속한 클래스를 검사하기 위하여 instanceof 연산자를 사용할 수 있다.

문 15. 다음 C++ 프로그램은 컴파일 시 오류가 발생한다. 오류의 원인에 대한 설명으로 옳은 것은?

```
#include <iostream>

using namespace std;

class Base {
protected :
    int a, b;
public :
    void setab(int n, int m) { a = n; b = m; }
};

class Derived : protected Base {
    int c;
public :
    void setc(int n) { c = n; }
    void showabc() {
        cout << a <<' '<< b <<' '<< c << endl;
    }
};

int main() {
    Derived ob;
    ob.setab(1, 2);
    ob.setc(3);
    ob.showabc();
    return 0;
}
```

① Derived의 생성자가 정의되어 있지 않음에도 불구하고 main 함수에서 Derived의 객체를 생성하였기 때문
② Derived의 setab 함수가 보호멤버(protected member)임에도 불구하고 main 함수에서 사용하였기 때문
③ a와 b는 Derived에서 전용멤버(private member)임에도 불구하고 showabc 함수에서 사용하였기 때문
④ 클래스 상속 시 보호(protected)로 상속할 수 없음에도 불구하고 Derived를 보호로 상속하였기 때문

문 16. 다음 문맥무관문법(context-free grammar)에서 문장 a*(a+a)의 좌파스(left parse)와 우파스(right parse)로 옳은 것은?

1. E → E + T
2. E → E * T
3. E → T
4. T → (E)
5. T → a

	좌파스	우파스
①	2 3 5 4 1 3 5 5	2 4 1 5 3 5 3 5
②	2 3 5 4 1 3 5 5	5 3 5 3 5 1 4 2
③	2 4 1 5 3 5 3 5	2 3 5 4 1 3 5 5
④	5 3 5 3 5 1 4 2	2 3 5 4 1 3 5 5

문 17. Java 언어에서 스레드 객체의 생성 방법으로 옳지 않은 것은?

① new Thread();
② new Thread() { public void run() {} };
③ new Thread(new Runnable() { public void run() {} });
④ new Runnable() { public void run() {} };

문 18. 다음 Java 프로그램의 실행 결과로 옳은 것은?

```
public class FourTimes {
    private static int j = 0;

    public static boolean methodB(int k) {
        j += k;
        return true;
    }

    public static void methodA(int i) {
        boolean b;
        b = (i<10 | methodB(4)) &&
            (i<10 || methodB(4));
    }

    static void methodC() {
        try {
            methodD();
            methodB(4);
        } catch (ArithmeticException e) {
            methodB(4);
        } catch (NullPointerException e) {
            methodB(4);
        } catch (Exception e) {
            methodB(4);
        } finally {
            methodB(4);
        }
    }

    static void methodD() {
        throw new NullPointerException();
    }

    public static void main(String[ ] args) {
        methodA(0);
        try {
            methodC();
        } catch (Exception e) {
            methodA(10);
        }
        System.out.println(j);
    }
}
```

① 8
② 12
③ 16
④ 20

문 19. 다음 Java 프로그램에 대한 설명으로 옳지 않은 것은?

```
import java.util.*;
class Guest {
    private String _name;
    public Guest(String name) { _name = name; }
    public String getName() { return _name; }
}
class GuestList extends Vector<Guest> {
    public GuestList() { super(); }
    public Guest getAt(int index) {
        return super.get(index);
    }
    public void addTail(Guest ptr) { super.add(ptr); }
    public void removeGuest(String name) {
        Guest tmp;
        for (int i = 0; i< this.elementCount; i++) {
            tmp = this.elementAt(i);
            if (name.equals(tmp.getName()))
                super.removeElementAt(i);
        }
    }
}
public class GuestMan {
    public static void main(String[ ] args) {
        GuestList guests = new GuestList();
        guests.addTail(new Guest("Jennifer"));
        guests.addTail(new Guest("Susan"));
        guests.addTail(new Guest("Mary"));
        guests.removeGuest("Jennifer");
        Iterator<Guest> i = guests.iterator();
        while (i.hasNext())
            System.out.print(i.next().getName() + " ");
    }
}
```

① 출력결과는 “Susan Mary ”이다.
② 범용적인 Vector 클래스를 직접 사용하는 대신 GuestList를 정의하여 특별한 용도를 나타내고 있다.
③ 컨테이너 객체 guests에 저장될 수 있는 Guest 객체 수는 미리 정해져 있지 않다.
④ 컨테이너 객체 guests는 매개변수적 다형 리스트(parametric polymorphic list)로 사용할 수 있다.

문 20. LR 구문분석에서 모호한 문법을 사용할 때 충돌(conflict)을 해결하는 방법으로 옳지 않은 것은?

① 액션테이블(action table)의 기억 공간을 축소하여 충돌을 해결할 수 있다.
② 이동-축약(shift-reduce) 충돌의 경우에는 이동을 우선적으로 선택함으로써 충돌을 해결할 수 있다.
③ 연산자 우선순위(precedence)를 고려하여 충돌을 해결할 수 있다.
④ 연산자 결합법칙(associativity)을 고려하여 충돌을 해결할 수 있다.