프로그래밍언어론

문 1. 다음 C 프로그램에서 Add 함수로 b를 전달하기 위하여 사용된 매개변수 전달(parameter passing) 방법은?

```
#include <stdio.h>

int Add(int size, int a[]) {
  int i, sum = 0;
  for (i=0; i<size; i++) sum += a[i];
  a[0] = 20;
  return sum;
}

void main() {
  int i, b[3];
  b[0] = 2; b[1] = 4; b[2] = 8;
  printf("the sum = %d\n", Add(3, b));
  for (i=0; i<3; i++) printf("%d", b[i]);
  printf("\n");
}
```

① Call by value ② Call by value result
③ Call by name ④ Call by reference

문 2. 객체지향 프로그래밍(object-oriented programming) 측면에서 C++와 Java를 비교한 설명으로 옳지 않은 것은?

① Java는 C++에 비해 객체지향 프로그래밍의 뜻에 보다 충실한 언어이다. 즉, 프로그램의 중심 단위가 C++에서는 함수(function)이나, Java에서는 클래스(class)이다.

② C++와 Java 모두 C 언어로부터 파생된 것들이다. 특히 C++는 단지 C 언어에 클래스(class) 개념을 추가시킨 명령형 언어(imperative language)로서 출발하였기에, 처음 이름이 클래스를 가진 C(C with classes)였다.

③ C++가 다중 상속(multiple inheritance)을 지원하는 데 반해, Java는 단일 상속(single inheritance)만 지원한다. Java에서는 다중 상속의 효과를 거두기 위해 인터페이스(interface)를 사용한다.

④ Java는 C++에 비해 객체지향 프로그램의 개념을 보다 충실히 구현함으로써 프로그램의 실행 속도도 더 빨라 현실에서 점차 C++의 적용 분야를 대치해 나가고 있다.

문 3. 다음 C 프로그램의 수행 결과는?

```
#include <stdio.h>
void main() {
    float a = 3/2;
    float b = 3.0/2;
    int c = (int)b;
    printf("%7.3f%7.3f%3d\n", a, b, c);
}
```

① 1.000 1.500 1 ② 1.000 1.000 1
③ 1.500 1.500 1 ④ 1.500 1.500 2

문 4. 프로그램의 구문 분석(syntax analysis)에 대한 설명으로 옳지 않은 것은?

① 좌파스(left parse) 또는 우파스(right parse)의 구성 여부에 따라 구문 분석 방법은 하향식(top-down) 방식과 상향식(bottom-up) 방식으로 구분된다.

② 올바른 문장에 대해 파스트리(parse tree) 또는 추상 구문 트리(abstract syntax tree)가 구성되면 추상 구문 트리가 파스트리에 비해 기억 공간이 효율적이다.

③ 하향식 구문 분석 방법에서 결정적 구문 분석(deterministic syntax analysis)을 위해 FIRST, FOLLOW를 이용한다.

④ LR 파싱 방법의 종류로서 CLR, LALR, 재귀 하강(recursive - descent) 파싱 방법이 있다.

문 5. 다음 Java 프로그램의 수행 결과는?

```
interface A {
  int a = 1;
  public int aa();
}
interface B {
  int a = 2;
  public int bb();
}

interface C extends A, B {
  int a = 3;
  public int cc();
}
class D implements C {
  public int aa() { return (5); }
  public int bb() { return (6); }
  public int cc() { return (7); }
}

class iTest {
  public static void main(String args[]) {
    A da = new D();
    C dc = new D();
    D dd = new D();

    System.out.println(da.a + dc.a + dd.a + dd.cc());
  }
}
```

① 10 ② 13
③ 14 ④ 16

문 6. 다음 Java 프로그램은 컴파일(compile) 도중 오류가 발생한다. 정상적으로 컴파일 되도록 하기 위한 조치로 옳지 않은 것은?

```
class Test {
    Test() { System.out.println("C()"); }
    Test(int n) { System.out.println("C(I)"); }      // ㉠
    void m() { System.out.println("()V"); }          // ㉡
    void m(long n) { System.out.println("(L)V"); }   // ㉢
    int m(long n) { System.out.println("(L)I");      // ㉣
                    return (int)n; }                 // ㉤
}
```

① ㉠, ㉡을 모두 제거 ② ㉡, ㉢을 모두 제거
③ ㉡, ㉣을 모두 제거 ④ ㉢, ㉣을 모두 제거

문 7. 예외 처리(exception handling)는 오늘날 현실적인 필요성으로 인하여 프로그래밍 언어 설계에서 점차 보편화되고 있다. 예외 처리 기능에 대한 설명으로 옳지 않은 것은?

① 프로그래밍 언어에 예외 처리 기능이 내장되면, 예외를 탐지하고 처리하기 위한 코드가 추가되어 그러한 기능이 없이 프로그래머가 예외를 처리하는 것보다 일반적으로는 프로그램이 길고 복잡해진다.

② 예외 전파(exception propagation) 기능을 이용하면 하나의 예외 처리기(exception handler)를 사용하여 여러 상이한 프로그램 단위에서 발생한 예외를 다룰 수 있으므로, 개발 비용을 절약할 수 있다.

③ 사용자가 작성한 예외 처리기가 예외를 처리하게 함으로써, 시스템에서 미리 정의된 포괄적이고 조악한 예외 처리나 메시지(message) 대신에 예외를 세밀하게 분류하여 사용자가 바라는 처리와 메시지를 기대할 수 있다.

④ 프로그래밍 언어에 예외 처리 기능이 포함되어 있다면, 지정된 예외를 검사하는 코드를 생성된 목적 코드(object code)에 삽입하도록 컴파일러가 설계된다.

문 8. 병행 시스템(concurrent system)에서 두 개의 프로세스 A와 B가 공용의 정수 자료 값 R에 동시에 접근하여 각각 1씩을 더해 준다고 할 때, R의 최종 값이 2만큼 증가한다는 것을 보장하지 못할 수도 있다. 이와 같이 병행 프로세스들이 공용 자원에 비배타적(non-exclusive)으로 접근하는 상황을 무엇이라 하는가?

① 경쟁 동기화(competition synchronization)
② 경쟁 조건(race condition)
③ 교착 상태(deadlock)
④ 협동 동기화(cooperative synchronization)

문 9. l-값(left-hand side value)과 r-값(right-hand side value)의 설명으로 옳은 것은?

① l-값과 r-값을 구별하기 위해서 ref를 사용하는 프로그래밍 언어는 Bliss이다.

② 연산자가 들어있는 수식에는 l-값은 있지만 r-값은 일반적으로 존재하지 않는다.

③ p가 포인터 변수인 경우, p의 r-값은 p가 지적하는 위치가 되며, l-값은 p 자신의 값이 들어있는 위치를 의미한다.

④ Algol 68의 변수 이름은 배정문 좌우 어느 위치에 존재하더라도 언제나 r-값을 의미한다.

문 10. 다음은 Java 바이트코드(bytecode)의 일부이다. 이 코드가 차례대로 실행되고 난 후 피연산자 스택(operand stack)의 꼭대기(top)에 저장되는 것은?

```
iconst_1
iconst_2
iconst_3
imul
iconst_4
iadd
imul
```

① -2 ② 4
③ 10 ④ 28

문 11. C 언어에서 int a[10]; 과 같이 선언하는 경우에 대한 설명으로 옳은 것은?

① a 배열이 필요로 하는 기억장소의 용량이 번역시간에 확정된다.
② a = 10; 과 같은 동작이 허용된다.
③ 프로그램 실행 중에 a 배열의 크기가 변한다.
④ a 배열이 저장되는 기억장소 내 주소는 컴파일러가 결정한다.

문 12. 다음의 4가지 C 함수들 가운데 그 의미가 나머지와 다른 것은?

①
```
int sum(int a[], int n) {
    int i, s = 0;
    for (i=0; i<n; i++)
        s += a[i];
    return s;
}
```

②
```
int sum(int *a, int n) {
    int i = 0, s = 0;
    while (i<n)
        s += a[i++];
    return s;
}
```

③
```
int sum(int a[], int n) {
    int i, s = 0;
    for (i=0; i<n; i++)
        s += *a++;
    return s;
}
```

④
```
int sum(int *a, int n) {
    int i = 0, s = 0;
    while (i<n)
        s += a[i]++;
    return s;
}
```

문 13. 객체지향 용어에 대한 설명으로 옳지 않은 것은?

① 소멸자(destructor) – 클래스의 객체가 종료될 때 자동으로 실행되는 메소드나 함수

② 생성자(constructor) – 클래스의 객체를 생성하는 메소드나 함수

③ 클래스(class) – 유사한 객체들을 묶어서 하나의 공통적인 특성을 표현한 것

④ 추상 메소드(abstract method) – 코드를 가지지 않고 시그너처(signature)만을 갖는 메소드

문 14. 바인딩(binding)에 대한 설명으로 옳은 것은?

① 언어 정의 시간(language definition time)에는 각 자료형(data type)이 표현될 수 있는 범위가 결정된다.

② 프로그램의 효율적인 실행을 목표로 하는 프로그래밍 언어는 바인딩을 주로 번역 시간(translation time)에 수행한다.

③ 언어 구현 시간(language implementation time)에 발생하는 바인딩을 최소화하면 그 언어는 호환성(portability)이 떨어진다.

④ 실행 시간(execution time) 바인딩을 위주로 하는 언어는 번역 시간 바인딩을 위주로 하는 언어보다 융통성(flexibility)이 떨어진다.

문 15. 프로그래밍 언어에서는 유연한 타입 체계를 제공하기 위해 여러 형태의 다형성(polymorphism)을 제공한다. 다음 중 오버로딩(overloading)에 해당하는 것을 모두 고른 것은?

ㄱ. C++의 템플릿(template)

ㄴ. Java에서 상속받은 메소드(method)를 재정의하여 사용

ㄷ. C에서 1+1.0을 타입 오류 없이 계산

ㄹ. Java에서 1.0+1.0과 "a"+"b"의 +가 피연산자(operand)의 타입에 따라 다르게 계산

ㅁ. Java에서 PrintStream 클래스의 println 메소드 사용

① ㄱ, ㄹ ② ㄴ, ㅁ

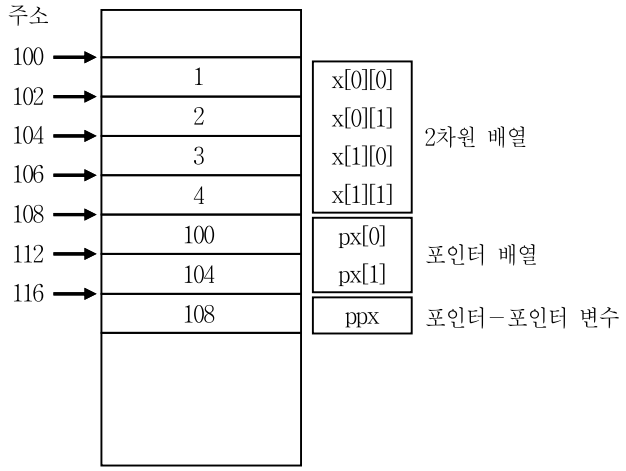
③ ㄷ, ㄹ ④ ㄹ, ㅁ

문 16. 다음 C 프로그램의 수행 결과는?(단, 배열 및 포인터 배열, 포인터-포인터 변수들이 메모리에 할당된 모습은 아래 그림과 같다고 가정한다)

```
#include <stdio.h>
void main() {
 short x[2][2] = {{1,2},{3,4}};
 short *px[2]; //포인터 배열
 short **ppx; //포인터-포인터 변수

 px[0] = x[0]; px[1] = x[1];
 ppx = px;

 printf("%10u %10u %10u \n", px[0], px[1], ppx);
 printf("%10u %10u %10u \n", px[0]+1, px[1]+1, ppx+1);
 printf("%10u %10u \n", *(++px[0]), *(++ppx));
}
```



① 100 104 108
102 106 112
2 104

② 100 104 108
102 106 112
1 100

③ 100 104 108
101 105 109
2 104

④ 100 104 108
101 105 109
1 100

문 17. 블록 구조(block structured) 언어로 작성된 다음 프로그램에 대하여 정적 영역 규칙(static scope rule)과 동적 영역 규칙(dynamic scope rule)이 각각 적용될 때 ㉠에서 프린트되는 b의 값은?

```
begin
    boolean b := true;
    procedure p
    begin
      print(b); -------- ㉠
    end p;
    begin
      boolean b := false;
      call p;
    end;
end;
```

	정적영역규칙	동적영역규칙
①	true	true
②	false	false
③	true	false
④	false	true

문 18. 다음 Java 프로그램의 수행 결과는?

```
public class C {
    private static int num = 0;

    protected static int get() {
        return num;
    }

    private static void add() {
        num++;
    }

    C() {
        C.add();
    }

    public static void main(String args[]) {
        for (int i=0; i<10; ++i)
            new C();
        System.out.println("The Value is " + C.get());
    }
}
```

① The Value is 1 ② The Value is 0
③ The Value is 10 ④ The Value is 9

문 19. 프로그래밍 패러다임(paradigm)에 대한 설명으로 옳지 않은 것은?

① 명령형 언어(imperative language)는 폰 노이만(von Neumann) 구조에 근거하였기 때문에 프로그램이 정확하게 실행된다는 것을 증명하기가 비교적 용이하다.
② 논리형 언어(logic language)는 기호 논리에 근거하였기 때문에 if 문이나 for 문과 같은 제어문을 사용하지 않더라도 프로그래밍이 가능하다.
③ 순수 함수형 언어(pure functional language)는 변수의 개념을 갖고 있지 않아 배정문을 사용하지 않고도 프로그래밍이 가능하다.
④ 객체지향 언어(object-oriented language)는 실세계에서 객체의 상호작용을 반영하기 위한 아이디어로 시작되어 코드의 재사용과 수정 용이성 증진에 매우 효과적인 방법으로 알려져 있다.

문 20. 다음과 같은 문법으로 주어진 수식을 계산할 때, 결과 값으로 옳은 것은? (단, 시작 심볼은 E이다)

```
E → T * E | T
T → F - T | F
F → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

주어진 수식 : 3 - 2 * 5 - 2 - 1
```

① -10 ② 4
③ 2 ④ -5