

프로그래밍언어론

2017년도 국가공무원 5급(기술) 공개경쟁채용 제2차시험

응시번호 : 성명 :

제 1 문. 배열은 동일한 데이터 형을 가진 값들을 모아놓은 것이다. 배열에 관하여 다음 물음에 답하시오. (총 10점)

1) C 언어와 Java 언어에서 각각 다음과 같이 1차원 배열을 생성할 때, 생성되는 메모리 구조의 차이를 설명하시오. (5점)

<C 코드>	<Java 코드>
int score[3] = {10, 20, 30};	int[] score = new int[3];
	score[0] = 10; score[1] = 20; score[2] = 30;

2) 다음 C 프로그램의 실행 결과와 그 이유를 설명하시오. (단, 배열의 시작 주소는 0x10 번지라고 가정한다) (5점)

```
void main(void) {
    char arr[3][3] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'};
    int i, j;
    i = j = 1;

    printf("1번 = %p\n", arr + i);
    printf("2번 = %p\n", arr[i] + j);
    printf("3번 = %c\n", *(arr[i++] + j));
    printf("4번 = %p\n", *(arr + i) + j);
    printf("5번 = %c\n", (*(arr + i) + j));
}
```

제 2 문. 다음 물음에 답하시오. (총 10점)

1) 다음 코드가 C 언어와 Java 언어에서 컴파일 또는 실행될 때, 결과의 차이와 그 이유를 설명하시오. (5점)

```
a = 0;
while (a = 10) a++;
```

2) 다음 C 코드와 Java 코드에서 강제 형변환(type casting)을 했을 때, 각각 어떻게 처리되는지 설명하시오. (5점)

<C 코드>	<Java 코드>
int arr[10];	Object ob = new String("a");
void* p = arr;	...
char* str;	Integer myint = (Integer)ob;
...	
str = (char*)p;	

제 3 문. Python 언어로 작성한 아래 f(n)과 g(n) 함수는 피보나치(Fibonacci) 수를 구하는 함수로서 실행의미가 동일하다. 즉, 모든 정수 n에 대해서 f(n)과 g(n)의 실행결과가 같다. 여기서 f(n) 함수는 재귀적으로 작성된 함수형 패러다임이고, g(n) 함수는 반복문으로 작성된 명령형 패러다임이다. 다음 물음에 답하시오.
(총 12점)

```
def f(n):
    if n <= 0: return 0
    elif n == 1: return 1
    else: return f(n-1) + f(n-2)
def g(n):
    a, b = 1, 0
    while n > 0:
        a, b = a + b, a
        n = n - 1
    return b
```

- 1) 실행시간 및 사용공간을 기준으로 두 함수의 차이점을 설명하시오. (5점)
- 2) 1)의 답에 기초하여 f(n) 함수 또는 g(n) 함수를 실행시간 및 사용공간을 줄일 수 있도록 변경하시오. (7점)

제 4 문. 다음 프로그램은 난수를 생성하여 스택에 저장하는 쓰레드(thread)와 저장된 난수를 꺼내서 출력하는 쓰레드가 동시에 수행되는 병행(concurrent) 프로그램이다. 물음에 답하시오.
(총 18점)

```
public class Main {
    public static void main(String[] args) {
        Stack stack = new Stack(10);
        Thread p = new Producer(stack);
        Thread c = new Consumer(stack);

        p.start();
        c.start();
    }
}

class Producer extends Thread {
    Stack stack;
    Producer(Stack s) {
        this.stack = s;
    }
    public void run() {
        while(true) {
            int r = (int) (Math.random()*500);
            try {
                Thread.sleep(r);
            }
            catch (Exception e) {}
            System.out.println("put "+r);
            stack.putNumber(r);
        }
    }
}
```

```

class Consumer extends Thread {
    Stack stack;
    Consumer(Stack s) {
        this.stack = s;
    }
    public void run() {
        while(true) {
            try {
                Thread.sleep((int)(Math.random()*500));
            }
            catch (Exception e) {}
            System.out.println("get " + stack.getNumber());
        }
    }
}

class Stack {
    int numbers[];
    int top = 0;
    int size = 0;
    Stack(int n) {
        size = n;
        numbers = new int[n];
    }

    public void putNumber(int n) {
        if (size == top) return;
        numbers[top] = n;
        top++;
    }
}

```

```

public int getNumber() {
    if (top == 0) return -1;
    top--;
    int n = numbers[top];
    return n;
}

class BufferFullException extends Exception { }

```

- 1) 버퍼에 빈 공간이 없는 꽉 찬 상태에서 putNumber 메소드가 호출되면 BufferFullException을 발생시키도록 프로그램을 수정하십시오. (putNumber 메소드의 타입은 public void putNumber(int n) throws BufferFullException이 된다) 또한 putNumber 메소드가 해당 예외상황을 발생시키면, “Buffer가 꽉 찼습니다” 메시지를 출력하고, 기존의 난수를 버퍼에 추가하는 반복 작업을 재개하도록 예외 처리 코드를 추가하십시오. (6점)
- 2) 위 Java 프로그램은 스택에 넣은 난수가 출력되기 전에 새로운 난수로 덮여 켜지는 경우가 발생할 수 있다. 이러한 문제가 발생되지 않도록 프로그램을 수정하십시오. (6점)
- 3) 2)에서 수정한 프로그램에 대해서 getNumber가 호출되었으나, 버퍼가 비어있을 경우 -1을 반환하는 것이 아니라, 새로운 난수가 생성될 때까지 해당 스레드가 기다렸다가 새로 추가된 난수를 반환하도록 Stack 클래스를 수정하십시오. (6점)

인사혁신처 시험출제과장