2022학년도 중등학교교사 임용후보자 선정경쟁시험

정보 · 컴퓨터

수험 번호 : (　　)　　**성　명 : (　　)**

제1차 시험	3교시 전공B	11문항 40점	시험 시간 90분

○ 문제지 전체 면수가 맞는지 확인하시오.
○ 모든 문항에는 배점이 표시되어 있습니다.

1. 다음은 고시 연도가 서로 다른 정보과 교육과정의 설명 자료를 순서 없이 제시한 것이다. (가)의 (　) 안에 들어갈 용어를 쓰고, 교육과정이 고시된 순서대로 (가)~(라)를 배열하시오. [2점]

(가)	정보 교과에서 추구하는 역량으로 '정보문화소양', '컴퓨팅 사고력', '협력적 문제해결력'이 제시되었고, '정보문화소양'의 하위 요소는 '정보윤리의식', (　　), '정보기술활용능력'이 있다.
(나)	중학교와 고등학교의 과목 명칭이 처음으로 정보(Informatics)로 통일되었고, 교과의 학습 내용이 컴퓨터 과학의 기본 개념과 원리로 구성되었다.
(다)	인공지능의 기본 개념과 원리, 기술을 활용하여 실생활 및 다양한 분야의 문제를 창의적으로 해결할 수 있는 기초 소양을 기르기 위한 선택 과목이 고시되었다.
(라)	중학교 과목의 목표에 계산적 사고(computational thinking)가 제시되었고, 이전의 내용 요소 1단계, 2단계, 3단계가 하나의 단계로 통합되었다.

2. <조건>을 고려하여 R 릴레이션에 존재하는 함수적 종속성 중 1개와, R이 만족하는 가장 높은 정규형을 각각 쓰시오. [2점]

R 릴레이션(기본키 : A)

A	B	C	D
1번	원숭이	유럽	지중해
2번	토끼	아시아	태평양
3번	원숭이	아시아	태평양
4번	토끼	아시아	대서양
5번	코끼리	아시아	대서양
6번	코끼리	유럽	지중해
7번	코끼리	유럽	지중해

<조 건>

○ 함수적 종속성 작성 시 자명한 함수적 종속성(B → B, BC → B 등)과 결정자(determinant)에 기본키가 포함된 함수적 종속성(A → D, AB → C 등)은 제외한다.
○ 정규화 과정에서 폐포(closure)와 규준 커버(canonical cover)는 고려하지 않는다.

3. (가)는 예비 교사가 작성한 시범·실습 수업 계획이고, (나)는 지도 교사와 예비 교사의 대화이다. <작성 방법>에 따라 서술하시오. [4점]

(가)

○ 수업 목표

• 배열과 제어 구조를 활용하여 합계를 구하는 프로그램을 작성할 수 있다.

○ 수업 설계

수업 단계		교사 활동
도입	㉠	• 음식점 주인이 하루 총 매출을 POS 시스템으로 계산하는 동영상을 제시한다. • 실생활에서 합계를 구하는 프로그램이 사용되는 예를 발표하게 한다.
전개	문제 이해 및 분석	• 입력 값, 처리 과정, 출력 값을 발표하게 한다. • 배열과 제어 구조를 어떻게 활용할지 발표하게 한다.
	알고리즘 설계	• 배열과 제어 구조를 활용하여 합계를 구하는 프로그램의 순서도를 작성하도록 한다.
	프로그래밍	• 제어 구조로 배열을 처리하는 프로그램을 시범·실습하도록 한다. • 순서도를 보면서 합계를 구하는 프로그램을 작성하도록 한다.
정리	평가 및 정리	• 프로그램의 정상 작동 여부를 서로 확인하도록 하고, 소감을 발표하게 한다.

(나)

예비 교사 : (가)와 같이 프로그래밍 시범·실습 수업을 설계했습니다.

지도 교사 : 컴퓨팅 사고력 활동이 수업 설계에 잘 나타나 있네요. (가)의 '프로그래밍' 단계에서 학습 초기에는 교사가 제시한 프로그램의 코드를 (㉡)하고, 그 코드를 (㉢)하는 활동으로 프로그래밍의 기본 개념과 원리를 습득하도록 하세요. 특히, 실습 과정에서 학생의 반복 연습과 교사의 피드백이 학습을 강화하는데, 이는 (㉣) 학습 이론에 기반합니다.

<작성 방법>

○ 켈러(J. Keller)의 용어를 사용하여 괄호 안의 ㉠에 해당하는 용어를 쓸 것.

○ 2015 개정 정보과 교육과정(교육부 고시 제2020-236호)의 '교수·학습 방법 및 유의 사항'에 근거하여 괄호 안의 ㉡과 ㉢에 해당하는 활동을 각각 쓸 것.

○ 괄호 안의 ㉣에 해당하는 용어를 쓸 것.

4. (가)는 중학교 정보 과목의 프로그래밍 영역에 대한 수업을 설계하기 위한 개요이며, (나)는 수업 일지 중 일부이다. <작성 방법>에 따라 서술하시오. [4점]

(가)

○ 중단원 : 3. 프로그래밍

○ 학습 대상 : 중학교 1학년

○ 학습 목표

1. 순차, 선택, 반복의 원리를 설명할 수 있다.
2. 순차, 선택, 반복의 개념을 적용하여 프로그램을 작성할 수 있다.

(나)

수업의 도입에서는 ㉠ 실생활에서 활용되고 있는 순차, 선택, 반복 구조에 대한 각각의 사례를 제시하여, 제어 구조에 대한 개념적 지식을 실제 대상에 투영하고자 하였다.

본시 학습에서는 순차, 선택, 반복 구조에 대한 개념을 설명하고 ㉡ 블록 기반 프로그래밍 언어를 사용하여 제어 구조의 특징을 시각적으로 구현하여 탐색하도록 하였다. 학생들은 프로그래밍을 수행하면서 제어 구조의 명령 실행 과정이 어떻게 다른지를 이해하고 효율적인 프로그램을 작성하도록 하였다. 프로그래밍을 통해 학생들은 (㉢)와/과 (㉣)을/를 구분하는 경계를 없앨 수 있었으며, 이는 직관적인 예상을 표면화하도록 도와주는 역할을 하였다. 학생들은 제어 구조를 이해하는 과정에서 동료교수법을 적용하여 결과와 과정을 공유하도록 하였다.

수업의 정리에서 학생들은 자신의 사고 과정에 대해 서로 이야기하면서 학습자 간 유의미한 상호작용과 구성원 각자의 역할을 책임감 있게 수행하는 모습을 확인하였다.

다음 차시 활동으로 실생활 문제를 해결할 수 있는 소프트웨어 설계와 개발을 위해 학습자 2명이 각각 ㉤ 드라이버와 내비게이터 역할을 수행하는 프로그래밍 방식인 (㉥)을/를 소개하며 수업을 마무리하였다.

<작성 방법>

○ 밑줄 친 ㉠과 ㉡이 페퍼트(S. Papert)의 구성주의(constructionism) 이론을 토대로 구성된 활동이라고 할 때, 괄호 안의 ㉢과 ㉣에 해당하는 용어를 쓸 것.

○ 괄호 안의 ㉥에 해당하는 용어를 쓰고, 밑줄 친 ㉤의 역할을 서술할 것.

5. (가)는 2015 개정 정보과 교육과정(교육부 고시 제2020-236호)에 근거하여 작성한 '프로그래밍' 영역의 평가 계획 중 일부이다. (나)는 프로그래밍 수업에서 학생 A의 답안과 교사의 답안 예시이다. <작성 방법>에 따라 서술하시오. [4점]

(가)

○ 관찰 평가 체크리스트

항목	평가 항목	평가 결과 (예 / 아니요)
1	(㉠)을/를 정의하고 활용하는 프로그램을 작성할 수 있는가?	
2	전역 변수와 지역 변수의 (㉡)을/를 구분할 수 있는가?	
3	값을 반환하는 (㉠)을/를 호출하여 활용할 수 있는가?	

(나)

○ 학생 A의 답안

```
#include <stdio.h>
void add(int a) {
    a = a + step;
}
int main(void) {
    int x = 3, y = 0;
    int step = 5;
    y = add(x);
    printf("%d", y);
}
```

○ 교사의 답안 예시

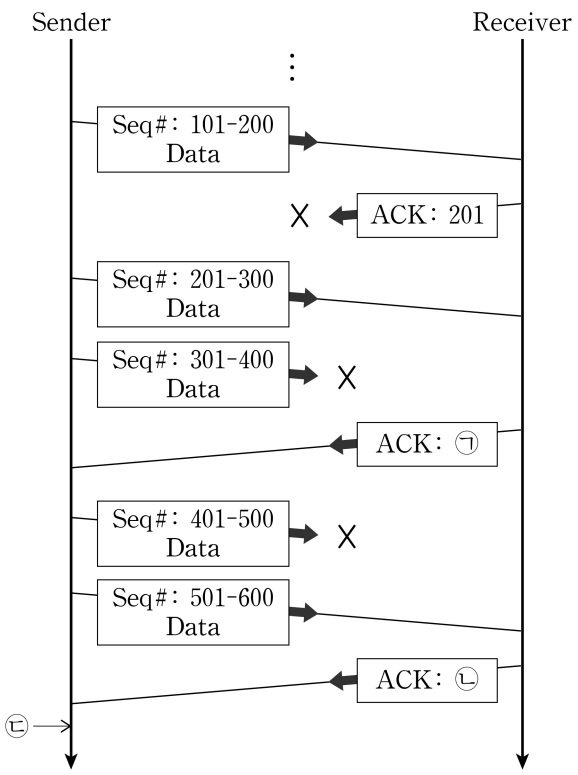
```
#include <stdio.h>
int step = 5;
int add(int a) {
    a = a + step;
    return a;
}
int main(void) {
    int x = 3, y = 0;
    y = add(x);
    printf("%d", y);
}
```

<작성 방법>

○ (나)를 참조하여 괄호 안의 ㉠과 ㉡에 들어갈 용어를 쓸 것.
○ (가)의 관찰 평가 체크리스트 2, 3번 항목에 따라 학생 A의 답안에 대한 평가 결과를 각각 쓰고, 학생 A의 답안 중 해당 코드 부분과 관련지어 교사의 피드백을 서술할 것.

6. (가)는 TCP(Transmission Control Protocol) 세그먼트(segment)의 전송 과정을 도식화한 것이다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

(가)



<조 건>

○ (가)에서 Seq#은 세그먼트에 포함된 데이터의 순서 번호(sequence number)를 나타낸다.
○ (가)에서 X 표시는 세그먼트가 손실되었음을 뜻한다.
○ MSS(Maximum Segment Size)는 100바이트이다.
○ TCP의 흐름 제어와 Selective ACK는 고려하지 않는다.
○ 수신된 세그먼트는 오류가 없다고 가정한다.

<작성 방법>

○ ㉠과 ㉡에 들어갈 숫자를 순서대로 쓸 것.
○ ㉢ 시점에서 timeout이 발생할 때 전송되는 세그먼트가 무엇인지 쓰고, 그 이유를 서술할 것.

7. (가)는 메인 메모리(main memory)에 적재된 데이터를 나타내고, (나)는 직접 사상(direct mapping)을 사용하는 캐시 메모리에 적재된 내용을 나타낸다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

(가)

메모리 주소

태그	라인	단어	데이터
0	00	0	A0A1
0	01	0	A0B1
0	10	0	A0C1
0	11	0	A0D1
1	00	0	A0E1
1	01	0	A0F1
1	10	0	B0A1
1	11	0	B0B1
1비트	2비트	1비트	16비트

(나)

라인	태그	데이터
00	1	A0E1
01	1	A0F1
10	0	A0C1
11	0	A0D1
2비트	1비트	16비트

<조 건>

○ 메모리 주소는 총 4비트로 아래와 같이 구성된다.

MSB	태그 필드	라인 필드	단어 필드	LSB
	1비트	2비트	1비트	

○ 메모리 주소는 바이트 단위로 지정되며, 단어(word)의 길이(length)는 1바이트이다.

○ 메인 메모리와 캐시 메모리 사이의 데이터 전송은 블록 단위로 이루어지고, 하나의 블록은 메모리 주소에서 태그 필드와 라인 필드가 같은 데이터 2바이트로 구성된다.

○ (가)와 (나)에서 데이터 값은 16진수이고, 비트 수를 제외한 나머지 숫자는 모두 2진수이다.

<작성 방법>

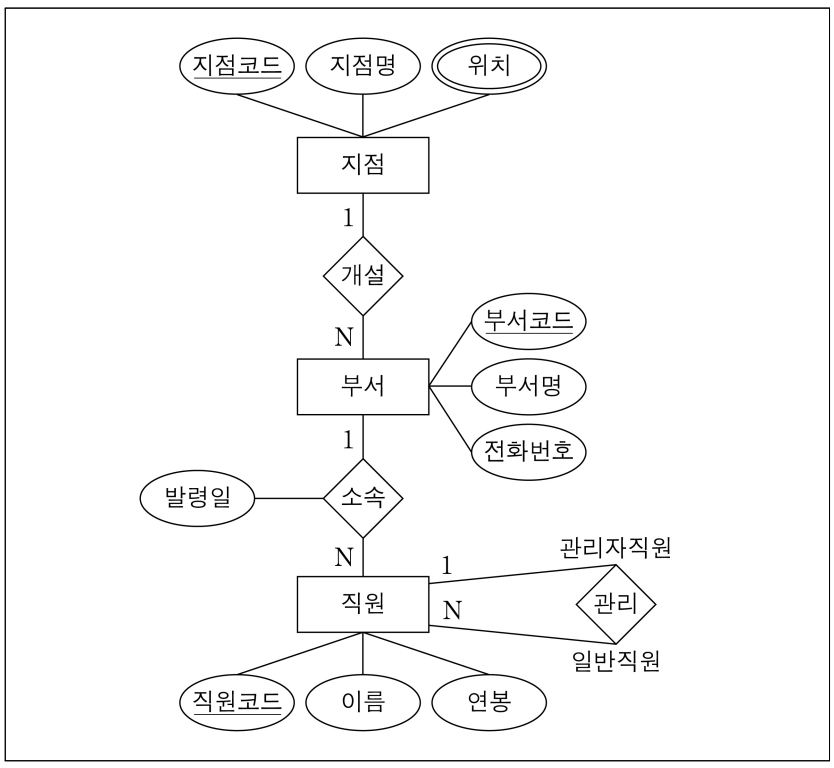
○ CPU가 메모리 주소 0010을 읽고 난 뒤 갱신된 캐시 메모리의 내용(라인, 태그, 데이터 포함)을 쓸 것.

○ 태그 필드가 1비트, 라인 필드가 2비트일 때 캐시 메모리에 저장할 수 있는 데이터는 총 8바이트이다. 태그 필드가 2비트, 라인 필드가 1비트로 변경된 시스템에서는 캐시 메모리에 저장할 수 있는 데이터가 총 몇 바이트인지 쓸 것.

○ 직접 사상 캐시 메모리가 연관 사상(associative mapping) 캐시 메모리와는 다르게 LRU(Least Recently Used)와 같은 캐시 엔트리 교체 알고리즘이 필요하지 <u>않은</u> 이유를 쓸 것.

8. (가)의 E-R 다이어그램을 릴레이션으로 변환하고자 한다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

(가)



<조 건>

○ (가)를 릴레이션으로 변환 시 데이터의 불필요한 중복이 발생하지 않도록 최소한의 릴레이션과 속성만으로 변환한다.

○ 릴레이션은 아래와 같은 형식으로 작성한다. 형식에서 PK와 FK는 각각 기본키와 외래키를 의미하며, 화살표는 참조하는 속성에서 참조되는 속성으로 향한다.

A 릴레이션

$속성_1$(PK)	$속성_2$	$속성_3$	…	$속성_n$

B 릴레이션

$속성_1$(PK)	$속성_2$(PK)	$속성_3$	…	$속성_m$(FK)

○ 정규화(normalization)는 고려하지 않는다.

<작성 방법>

○ '부서' 개체(entity) 타입으로부터 변환되는 릴레이션의 차수(degree)를 쓸 것.

○ '지점' 개체 타입으로부터 변환되는 릴레이션을 <조건>을 참조하여 작성할 것.

○ '직원' 개체 타입으로부터 변환되는 릴레이션을 <조건>을 참조하여 작성할 것.

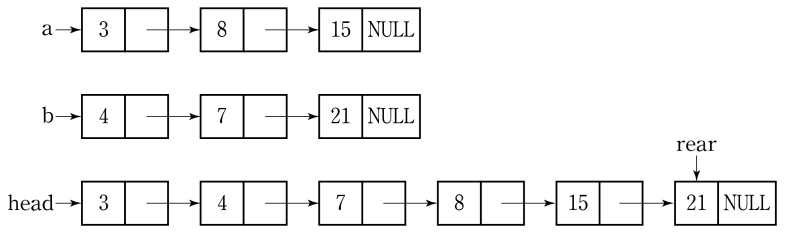
한국교육과정평가원

9. 다음은 두 개의 정렬된 리스트를 합병하여 하나의 정렬된 리스트로 만드는 C 프로그램의 일부이다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

```
typedef struct node * NODE;
struct node {
    int id;
    NODE next;
};
NODE mergelist(NODE a, NODE b){
   NODE head = NULL, rear = NULL;
   while(a != NULL && b != NULL) {
      if(rear == NULL){
         if(a->id < b->id){
            [        ㉠        ]
            a = a->next;
         } else {
            [        ㉡        ]
            b = b->next;
         }
      } else {
         if(a->id < b->id){
            rear->next = a;
            rear = rear->next;
            a = a->next;
         } else {
            rear->next = b;
            rear = rear->next;
            b = b->next;
         }
      }
   }
   if(a == NULL && b != NULL) {
      [        ㉢        ]
   } else if(a != NULL && b == NULL) {
      [        ㉣        ]
   }
   return head;
}
```

<조 건>

○ mergelist()는 2개의 정렬된 리스트를 받아 하나의 정렬된 리스트로 변환하여 반환한다. 예를 들어 정렬된 자료 (3, 8, 15)가 리스트 a이고 (4, 7, 21)이 리스트 b라면 합병 결과 리스트 head는 다음과 같다. (rear는 리스트의 끝을 가리킨다.)



<작성 방법>

○ mergelist() 함수가 올바르게 동작하도록 ㉠과 ㉡에 들어갈 코드를 순서대로 쓸 것.

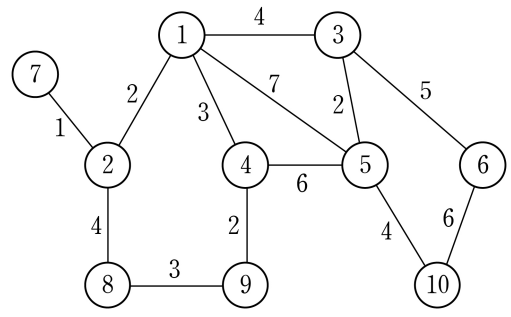
○ mergelist() 함수가 올바르게 동작하도록 ㉢과 ㉣에 들어갈 코드를 순서대로 쓸 것.

10. (가)는 그래프에서 최소비용신장트리를 구하는 알고리즘이고, (나)는 이 알고리즘을 적용할 대상 그래프이다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

(가)

```
T = { };
TV = {1};       /* 시작정점을 1로 한다 */
initnear();     /* 배열 near를 초기화함 */
near[1] = 0;    /* 시작점이 TV에 속하므로 이를 배제함 */
while(size(T) < n-1) {
   v = find(); /* TV에 속하지 않는 정점(near[i] ≠ 0)
                  중 (i, near[i])의 비용이 최소인 정점을
                  찾아 반환함 */
 ㉠ add((v, near[v]), T);
   vadd(v, TV);
   near[v] = 0;       /* 선택된 정점을 배제함 */
 ㉡ modifynear(v);    /* v가 TV에 속함에 따라 배열 near
                         값 수정 */
}
print(T);             /* 최소비용간선 출력 */
```

(나)



<조 건>

- ○ T는 현재 시점의 신장트리간선의 집합이고, TV는 현재 시점의 신장트리정점의 집합이다.
- ○ (나) 그래프의 각 간선에 표시된 값은 해당 간선의 비용이다.
- ○ 배열 near의 항목 수는 대상 그래프의 정점의 개수 n이다.
- ○ 그래프에 속한 각각의 정점 i에 대해 배열 항목 near[i]는 TV에 속한 정점 중 정점 i와 최소 비용으로 연결되어 있는 정점의 번호 값을 갖는다. 단, 정점 i가 TV에 속한 정점일 경우 near[i]는 0이고, TV에 속한 정점 중에 정점 i와 직접 연결된 정점이 없으면 near[i]는 시작 정점의 번호인 1로 설정된다.
- ○ find()는 near[i]의 값이 0이 아닌 정점 중 (i, near[i])의 비용이 최소인 정점을 찾아 반환한다. 간선이 존재하지 않는 두 정점 사이의 비용은 무한대로 가정한다.
- ○ (가) 알고리즘에서 near[1]의 값을 0으로 설정하는 부분까지 수행했을 때 배열 near의 항목 값들은 다음과 같이 설정된다.

인덱스(정점번호)	1	2	3	4	5	6	7	8	9	10
배열 near	0	1	1	1	1	1	1	1	1	1

- ○ 최소비용신장트리를 구해 가는 과정에서 새로운 정점 v를 TV에 추가하고 near[v]의 값을 0으로 변경하고 나면, modifynear(v)를 호출하여 TV에 속해 있지 않은 정점 i 각각에 대해 (i, near[i])보다 (i, v)의 비용이 더 작으면 near[i]의 값을 v로 변경한다. 예를 들어, 위에 제시된 상태에서 find()가 찾아낸 정점 2가 TV에 추가된다면, near[2]를 0으로 변경하고 modifynear(2)를 호출함으로써 TV에 새로 추가된 정점 2를 고려해 TV에 속해 있지 않은 정점 i(3 ~ 10) 중에 (i, near[i])보다 (i, 2)의 비용이 더 작은 것이 있으면 해당 near[i]의 값을 2로 변경한다.
- ○ 함수 size(T)는 간선집합 T에 속한 간선의 개수를 반환한다.
- ○ add((v, near[v]), T)는 간선 (v, near[v])를 집합 T에 추가하고, vadd(v, TV)는 정점 v를 집합 TV에 추가한다.

<작성 방법>

- ○ (나) 그래프에 (가) 알고리즘을 적용하였을 때 구하여지는 T에 포함된 모든 간선의 비용 합을 쓸 것.
- ○ (가)에서 밑줄 친 ㉠이 세 번째 수행될 때 T에 추가되는 간선을 쓸 것.
- ○ (가)에서 밑줄 친 ㉡이 세 번째 수행될 때 배열 near의 항목 값이 변경되는 정점들을 쓸 것.

한국교육과정평가원

11. (가)와 (나)는 시점 0에 단일 처리기 시스템 내에서 병행 수행되고 있는 모든 프로세스의 상태이다. (가)는 준비큐에서 CPU 할당을 대기하고 있는 프로세스들의 상태이고, (나)는 디스크 입출력 대기큐에서 자신이 요청한 입출력 작업이 끝나길 기다리고 있는 프로세스들의 상태이다. <조건>을 고려하여 <작성 방법>에 따라 서술하시오. [4점]

(가)

프로세스	CPU 버스트 시간
A	600
B	300
C	700

(나)

프로세스	요청한 입출력의 디스크 트랙 번호	입출력 서비스 종료 후 (가)의 준비큐에 진입하게 될 때 설정될 CPU 버스트 시간
D	380	400
E	310	700
F	350	100
G	720	800
H	340	500

<조 건>

- ○ (가)의 준비큐는 SJF(Shortest Job First : SPN(Shortest Process Next)이나 SPF(Shortest Process First)라고 불리기도 함.) 알고리즘에 의해 관리되고, (나)의 디스크 입출력 대기큐는 SSTF(Shortest Seek Time First) 알고리즘에 의해 관리된다.
- ○ 프로세스 A, B, C 중 하나가 SJF에 의해 선택되어 시점 0에 수행이 시작된다.
- ○ 시점 0에 프로세스 F의 입출력 요청에 대한 서비스가 진행 중이며, 이는 시점 50에 끝난다.
- ○ (가)와 (나)에서 'CPU 버스트 시간'은 해당 프로세스가 (가)의 준비큐에 진입한 이후부터 준비큐를 떠나게 될 때까지 할당받아 사용해야 할 총 CPU 시간을 의미한다.
- ○ 입출력 요청 각각을 처리하는 데 요구되는 입출력 서비스 시간은 '2×distance+200'이며, distance는 그 다음 입출력 서비스를 위해 헤드를 이동해 가야 하는 트랙 간 거리이다. 예를 들어, 시점 100에 입출력 서비스가 끝난 트랙의 번호가 370이고 그 다음 서비스 대상이 트랙 220상의 입출력 요청일 경우 distance의 값은 150(=370−220)이므로, 트랙 220상의 입출력 요청에 소요되는 입출력 서비스 시간은 500(=2×150+200)이 되고 해당 서비스의 종료 시점은 600이 된다.
- ○ 요청한 입출력 작업이 끝난 프로세스는 그 즉시 (가)의 준비큐에 추가된다.
- ○ 현재 시스템 내에 존재하는 프로세스들이 모두 종료될 때까지 생성되는 프로세스도 없고 디스크 입출력을 요청하는 프로세스도 없다고 가정한다.
- ○ 프로세스 스케줄링, 프로세스 문맥 교환 및 상태 전이, 인터럽트 발생 및 처리, 디스크 스케줄링 등에 소요되는 시간은 0으로 가정한다.

<작성 방법>

- ○ 시점 0부터 SJF에 의해 두 번째와 세 번째로 CPU를 할당받게 되는 프로세스를 순서대로 쓸 것.
- ○ 프로세스 F에 대한 입출력 서비스가 끝난 이후 SSTF에 의해 세 번째 서비스 대상으로 선정되는 입출력 요청의 트랙 번호와 해당 서비스가 종료되는 시점을 쓸 것.

<수고하셨습니다.>