문제 분석)

A의 위치, B의 위치, B의 최대 속도를 입력받는다.

현재 위치에서 B가 이길 수 있는 최소 속도 v\_min을 구한다.

v\_min이 B의 최대속도 보다 클 경우 다시 처음부터 입력 받는다.

A의 속도는 항상 2이다.

A와 B의 속도는 가속되지 않는다.

결승선을 먼저 벗어난 사람이 승리한다.

A, B의 속도는 정수로만 표현된다.

의사 코드 및 소스 코드 분석)

find\_min (Apos, Bpos, v) {

if (10 - Apos) / 2 <= (10 - Bpos) / v then

return find\_min(Apos, Bpos, v + 1)

else then

return v;

}

run() {

print “------------------------------------";

Apos, Bpos, v;

print "A의 위치를 입력하세요 :”;

get Apos;

printf "B의 위치를 입력하세요 :”;

get Bpos;

printf "B의 최대 속도를 입력하세요 :”;

get v;

set speed to find\_min(Apos, Bpos, 1)

print "B가 이길 수 있는 최소 속도는 %d 입니다.", speed;

if speed > v then

print "B의 최대속도는 %d이므로 B는 승리할 수 없습니다.", v;

run

}

main() {

printf "학번: 202300000 , 이름: 000";

run

}

int find\_min (int Apos, int Bpos, int v)

거리 = 속력 \* 시간 공식을 활용하여, A의 남은 거리에 A의 속도 (문제 조건에서 항상 2라고 명시)를 나눈 값과 B의 남은 거리에 B의 현재 속도를 나눈 값을 비교하여, A가 작을 경우는(목적지에 도달하는 시간이 더 짧을 경우) A가 B보다 더 빨리 결승점을 통과하므로 B의 속도를 현재보다 1 빠르게 하여 재귀를 수행한다.

해당 함수를 호출하는 경우 B의 속도는 항상 1일 때부터 호출한다.

void run()

프로그램의 전반적인 흐름을 구현한 함수이며, 속도와 위치 값들을 입력받고, 위에서 언급한 find\_min 함수를 통해 A와 B의 속도에 대한 비교 및 연산을 수행한 후 문제 조건에 따라서 출력한다.

결과)