OSHW#2 Deadlock

2016104142 이광원

1. 자료구조, 설정 과정

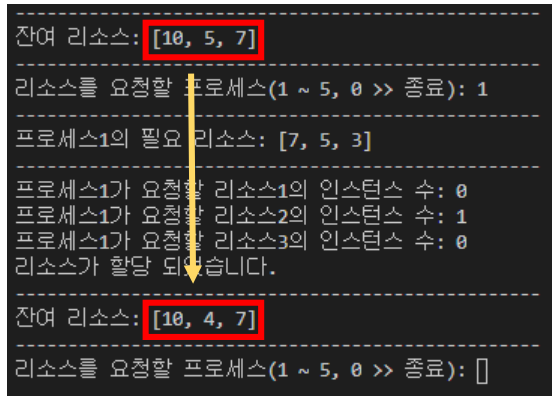
```python
#메인 프로그램
print("-" * 50)
n = int(input("프로세스의 수: "))   #프로세스의 수 n
m = int(input("리소스의 수: "))     #리소스의 수 m

Available = [0]*m   #리소스의 사용 가능한 인스턴스 수 저장
Instance = [0]*m    #리소스 별 최대 인스턴스 수 저장
Max = [[0 for j in range(m)] for i in range(n)] #프로세스가 사용할 최대 인스턴스 수 저장
Allocation = [[0 for j in range(m)] for i in range(n)]  #프로세스가 사용 중인 인스턴스 수 저장
Need = [[0 for j in range(m)] for i in range(n)]    #프로세스가 현재 필요한 인스턴스 수 저장

#초기 프로세스 및 리소스 설정 시퀀스
for i in range(m):
    print("-" * 50)
    x = int(input("리소스%d의 인스턴스 수: " % (i+1)))
    Instance[i] = x
    Available[i] = Instance[i]  #Available = Instance - Allocation(현재: 0)
    for j in range(n):
        Max[j][i] = int(input("프로세스%d이 요청할 리소스%d의 최대 인스턴스 수: " % (j+1, i+1)))
        Need[j][i] = Max[j][i]  #Need = Max - Allocation(현재: 0)
```

```
--------------------------------------------------
프로세스의 수: 5
리소스의 수: 3
--------------------------------------------------
리소스1의 인스턴스 수: 10
프로세스1이 요청할 리소스1의 최대 인스턴스 수: 7
프로세스2이 요청할 리소스1의 최대 인스턴스 수: 3
프로세스3이 요청할 리소스1의 최대 인스턴스 수: 9
프로세스4이 요청할 리소스1의 최대 인스턴스 수: 2
프로세스5이 요청할 리소스1의 최대 인스턴스 수: 4
--------------------------------------------------
리소스2의 인스턴스 수: 5
프로세스1이 요청할 리소스2의 최대 인스턴스 수: 5
프로세스2이 요청할 리소스2의 최대 인스턴스 수: 2
프로세스3이 요청할 리소스2의 최대 인스턴스 수: 0
프로세스4이 요청할 리소스2의 최대 인스턴스 수: 2
프로세스5이 요청할 리소스2의 최대 인스턴스 수: 3
--------------------------------------------------
리소스3의 인스턴스 수: 7
프로세스1이 요청할 리소스3의 최대 인스턴스 수: 3
프로세스2이 요청할 리소스3의 최대 인스턴스 수: 2
프로세스3이 요청할 리소스3의 최대 인스턴스 수: 2
프로세스4이 요청할 리소스3의 최대 인스턴스 수: 2
프로세스5이 요청할 리소스3의 최대 인스턴스 수: 3
```

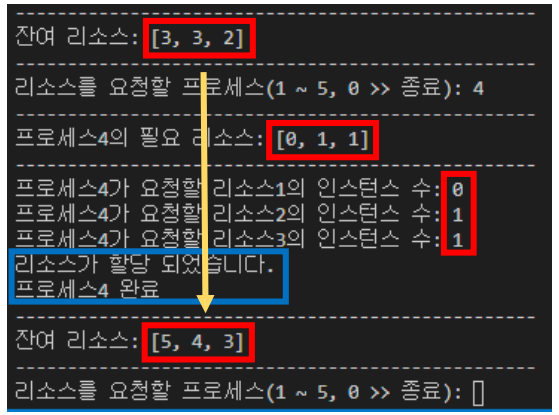
2. 리소스 할당/ 프로세스 완료

```
if Safety(n, m, Avl, Alc, Nd):  #Safety 테스트 실행
    Available, Allocation, Need = Avl, Alc, Nd
    print("리소스가 할당 되었습니다.")  #Deadlock이 발생하지 않으므로 리소스 할당

    if Done(m, Need[index]): #리소스 할당으로 프로세스가 종료 되는지 확인
        for i in range(m):  #프로세스 종료 시퀀스
            Available[i] += Allocation[index][i]    #잔여 리소스 증가
            Allocation[index][i] = 0    #사용 리소스 초기화(0)
        print("프로세스%d 완료" % (index+1))

    return Available, Allocation, Need  #변경된 리스트 반환
else:   #Deadlock이 발생
    print("요청이 Deadlock을 발생시킵니다.")
    return Available, Allocation, Need  #변경되지 않은 리스트 반환
```

a) 리소스 할당



b) 프로세스 완료



3-1. 요청 오류 처리(인스턴스 초과)

a) 요청 리소스 > 필요 리소스

```
if Request[i] > Need[index][i]: #요청 리소스가 필요 리소스 보다 큰 경우
    print("필요한 인스턴스보다 큰 값을 입력했습니다.")
    return Available, Allocation, Need  #변경되지 않은 리스트 반환
```

```
--------------------------------------------------
잔여 리소스: [10, 5, 7]
--------------------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료): 1
--------------------------------------------------
프로세스1의 필요 리소스: [7, 5, 3]
--------------------------------------------------
프로세스1가 요청할 리소스1의 인스턴스 수: 11
필요한 인스턴스보다 큰 값을 입력했습니다.
--------------------------------------------------
잔여 리소스: [10, 5, 7]
--------------------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료):
```

b) 요청 리소스 > 잔여 리소스

```
if Request[i] > Available[i]:   #요청 리소스가 잔여 리소스 보다 큰 경우
    print("인스턴스가 부족합니다.")
    return Available, Allocation, Need  #변경되지 않은 리스트 반환
```

```
--------------------------------------------------
잔여 리소스: [0, 5, 7]
--------------------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료): 3
--------------------------------------------------
프로세스3의 필요 리소스: [9, 0, 2]
--------------------------------------------------
프로세스3가 요청할 리소스1의 인스턴스 수: 3
인스턴스가 부족합니다.
--------------------------------------------------
잔여 리소스: [0, 5, 7]
--------------------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료):
```

3-2. 요청 오류 처리(Deadlock)

```python
#프로세스 실행에 Deadlock이 발생하는지 확인하는 알고리즘(n = 프로세스 수, m = 리소스 수)
def Safety(n, m, Available, Allocation, Need):
    Work = Available[:] #사용 가능한 인스턴스 수 저장
    Finish = [False]*n  #종료한 프로세스 확인

    while 1:
        count, done = 0, True   #조건 확인 변수
        for i in range(n):
            #프로세스가 종료하지 않았고, 잔여 리소스가 필요 인스턴스보다 큰 프로세스가 있다면
            if Finish[i] == False and Larger(m, Need[i], Work):
                for j in range(m):
                    Work[j] += Allocation[i][j] #사용 완료한 인스턴스를 반환
                Finish[i] = True    #프로세스 종료
                break   #다음 시퀀스로 이동
            else:
                count += 1
                done = done and Finish[i]   #모든 Finish 값이 True면 done이 True

        if count == n:  #조건에 만족하는 프로세스가 없는 경우
            if done: return True    #모든 프로세스가 종료한 상태(Deadlock이 발생하지 않음)
            else: return False      #종료되지 않은 프로세스가 있다(Deadlock 발생)
```

```
----------------------------------------
잔여 리소스: [0, 5, 5]
----------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료): 2
----------------------------------------
프로세스2의 필요 리소스: [3, 2, 2]
----------------------------------------
프로세스2가 요청할 리소스1의 인스턴스 수: 0
프로세스2가 요청할 리소스2의 인스턴스 수: 2
프로세스2가 요청할 리소스3의 인스턴스 수: 2
요청이 Deadlock을 발생시킵니다.
----------------------------------------
잔여 리소스: [0, 5, 5]
----------------------------------------
리소스를 요청할 프로세스(1 ~ 5, 0 >> 종료):
```

4. 프로그램 종료