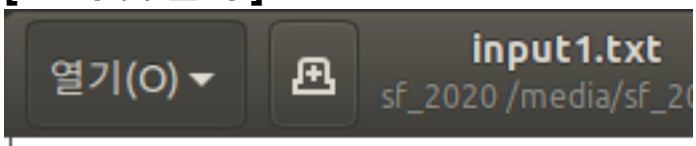
## [ 설계 및 구현 ]

1. input.txt에서 프로세스 개수, 리소스 개수와 리소스 유닛 개수를 읽어온다.
2. 프로세스 개수 x 리소스 개수 크기의 matrix – allocate, request-를 만든다.
3. 만들어진 2개의 행렬에 알맞은 데이터를 각각 input.txt 에서 읽어와 저장 한다.
4. 프로세스가 요구하는 모든 리소스의 리소스 유닛의 개수가 남아있는 리소스 유닛보다 작거나 같아야 그 프로세스는 unblocked이고 deadlock이 아니라고 말할 수 있다.
5. finish라는 어레이를 이용하여 리소스를 할당 받고 끝이 난, 즉 unblocked 프로세스를 표시한다.
6. 아직 끝나지 않은 프로세스(finish에 표시가 되어 있지 않은 프로세스)에 한해 4번의 조건을 부합하는지 확인한다.
7. 만약 조건에 부합하면 finish에 표시를 하고 할당 받았던 리소스를 반납한다.
8. 만약 조건에 부합하는 프로세스가 있었다면 change를 1로 변경한다.
9. 조건에 부합하는 프로세스가 발견되면 다시 finish가 표시되지 않은 첫 프로세스부터 조건에 부합하는지 확인한다.
10. 만약 모든 프로세스를 확인했는데 change가 1이 아니라면, 더 이상 unblocked process가 없이 남아 있는 프로세스는 모두 deadlock 프로세스이거나 모든 프로세스가 끝이 났다는 뜻이므로 detection을 마친다.
11. detection을 마치고 finish에 1이 표시되지 않은 프로세스는 deadlock process list에 출력한다.
12. 만약 모든 프로세스의 finish가 1이라면 blocked 프로세스가 없다는 뜻이 므로 시스템은 deadlock 상태가 아니다.
13. 만약 모든 프로세스의 finish가 1이진 않다면 blocked 프로세스가 1개 이상 있다는 뜻이므로 시스템은 deadlock 상태이다.

1. 직접 작성한 입력 파일을 사용하는 모드 (RANDOM_INPUT 0)
   난수를 생성하여 입력 파일을 만들어 사용하는 모드 (RANDOM_INPUT 1)
   RAND_RANGE: 프로세스와 리소스 개수의 범위 (0 ~ RAND_RANGE)
   UNIT_RANGE: 리소스 유닛의 개수의 범위(0 ~ UNIT_RANGE)
2. R_info를 통해 입력 파일의 첫 줄을 읽어오고 리소스 유닛의 개수를 저장한 resource pointer array을 반환 받는다.
3. R_info에서 읽은 프로세스 개수 x 리소스 유닛 종류의 크기를 가지는 행렬을 create_matrix를 이용하여 만든다. (allocate, request 총 2개의 행렬 만듦) 이 때 matrix는 동적 메모리 할당을 받고 실패 시 -1을 반환하며 종료한다. 만들어진 matrix는 0으로 초기화된다.
4. get_input을 통해 allocate, request 정보를 각각의 행렬에 저장한다. 만약 읽어 들인 정보가 명시된 유닛 종류보다 작으면 에러 메시지를  출력하고 -3을 반환하며 프로그램이 종료된다.
5. get_R_remain을 이용하여 남은 리소스 유닛 개수를 계산하고 만약 입력 받은 할당된 리소스 유닛의 개수가 총 리소스 유닛의 개수보다 크면 에러 메시지를 출력하고 -4를 return하며 프로그램이 종료된다.
6. 입력 받은 행렬을 출력한다.
7. detection에서 실제적인 deadlock detection이 진행된다.
8. 검사가 끝나고 allocate과 request 행렬을 다시 출력한다. 이 때 finish된 프로세스는 request와 allocate이 모두 0이다.
9. find_deadlock을 통해 deadlock detection의 결과를 출력한다.
10. free_matrix, free를 이용하여 동적 할당된 메모리를 반납한다.
11. 파일을 닫는다.

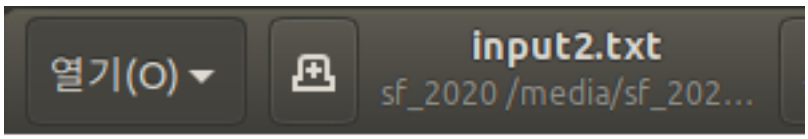
12. return

**[ 입력 및 출력 ]**




```
3, 3, 3, 2, 2
2, 1, 0
1, 0, 0
0, 1, 1
1, 1, 0
0, 2, 1
0, 0, 1
```

```
# of P: 3 # of R: 3
number of resource unit
3 2 2
request
1 1 0
0 2 1
0 0 1

allocate
2 1 0
1 0 0
0 1 1

Available resource
0 0 1

result: final allocation
2 1 0
1 0 0
0 0 0

result: remain request
1 1 0
0 2 1
0 0 0

Deadlocked process:
P0 P1
The system is in Deadlock state
```




```
2, 2, 3, 2
2, 0
1, 1
0, 1
1, 1
```

```
# of P: 2 # of R: 2
number of resource unit
3 2
request
0 1
1 1

allocate
2 0
1 1

Available resource
0 1

result: final allocation
0 0
1 1

result: remain request
0 0
1 1

Deadlocked process:
P1
The system is in Deadlock state
```

열기(O) input3.txt
sf_2020 /media/sf_20

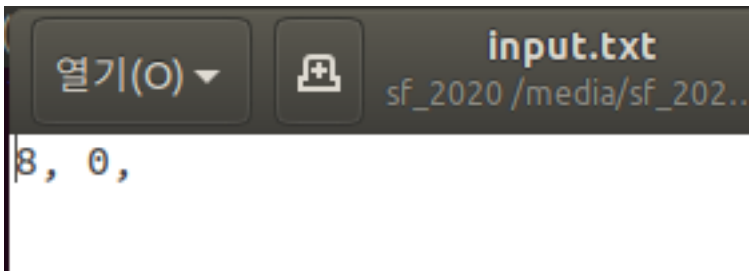
```
5, 3, 10, 5, 7
0, 1, 0
2, 0, 0
3, 0, 2
2, 1, 1
0, 0, 2
7, 4, 3
1, 2, 2
6, 0, 0
0, 1, 1
4, 3, 1
```

```
# of P: 5 # of R: 3
number of resource unit
10 5 7
request
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1

allocate
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Available resource
3 3 2

result: final allocation
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0

result: remain request
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0

Deadlocked process:
none
Not in Deadlock state
```

열기(O) input.txt sf_2020 /media/sf_202... 저장(S)

```
4, 10, 8, 2, 18, 15, 18, 8, 8, 20, 9, 17
8, 0, 5, 3, 17, 6, 2, 5, 2, 17
0, 1, 6, 6, 0, 0, 5, 12, 5, 0
0, 1, 5, 0, 1, 2, 1, 0, 1, 0
0, 0, 0, 6, 0, 0, 0, 3, 1, 0
0, 1, 0, 5, 1, 1, 6, 10, 6, 0
6, 1, 10, 7, 18, 0, 2, 4, 4, 6
1, 1, 3, 11, 6, 4, 6, 0, 3, 17
6, 0, 8, 3, 12, 1, 4, 10, 8, 16
```

```
# of P: 4 # of R: 10
number of resource unit
8 2 18 15 18 8 8 20 9 17
request
0 1 0 5 1 1 6 10 6 0
6 1 10 7 18 0 2 4 4 6
1 1 3 11 6 4 6 0 3 17
6 0 8 3 12 1 4 10 8 16

allocate
8 0 5 3 17 6 2 5 2 17
0 1 6 6 0 0 5 12 5 0
0 1 5 0 1 2 1 0 1 0
0 0 0 6 0 0 0 3 1 0

Available resource
0 0 2 0 0 0 0 0 0 0

result: final allocation
8 0 5 3 17 6 2 5 2 17
0 1 6 6 0 0 5 12 5 0
0 1 5 0 1 2 1 0 1 0
0 0 0 6 0 0 0 3 1 0

result: remain request
0 1 0 5 1 1 6 10 6 0
6 1 10 7 18 0 2 4 4 6
1 1 3 11 6 4 6 0 3 17
6 0 8 3 12 1 4 10 8 16

Deadlocked process:
P0 P1 P2 P3
The system is in Deadlock state
```




```
# of P: 8 # of R: 0
number of resource unit

request









allocate









Available resource

result: final allocation









result: remain request









Deadlocked process:
none
```

```
Not in Deadlock state
```

[ 결과 ]
Deadlock detection 기법을 통해 주어진 혹은 random으로 생성된 입력 파 일의 상황에서 deadlock 상태인지 아닌지를 파악할 수 있게 되었고 random으로 생성된 다양한 입력 파일을 통해 확인해본 결과 한 프로세스가 교착상태이면 시 스템 내 모든 프로세스가 교착상태일 가능성이 매우 높다는 것을 확인할 수 있었다.

/src
  —deadlock_detection.c
/data
  —input1.txt
  —input2.txt
  —input3.txt
  —input4.txt