1. 구현 내용 설명.

1) void input()

✓ 파일 입력을 받아 처리.

```
11
     void input(){
12
13
         FILE *fp;
14
         fp = fopen("input.txt","r");
          fscanf(fp, "%d %d", &num_process, &num_resource);
          for(int i=0;i<num_resource;i++){</pre>
              int resource_unit;
              fscanf(fp,"%d",&resource_unit);
              resourceUnits.push_back(resource_unit);
21
              sum_allocated.push_back(0);
23
          for(int i=0;i<num_process;i++){</pre>
25
              vector<int> elem;
              for(int j=0;j<num_resource;j++){</pre>
                  int num_allocated;
                  fscanf(fp,"%d",&num_allocated);
                  sum_allocated[j] += num_allocated;
30
                  elem.push back(num allocated);
              allocated.push_back(elem);
34
           for(int i=0;i<num_process;i++){</pre>
              vector<int> elem;
36
              for(int j=0;j<num_resource;j++){</pre>
                  int num_requested;
                  fscanf(fp,"%d",&num requested);
                  elem.push_back(num_requested);
40
              requested.push_back(elem);
          fclose(fp);
```

- 13~15: *num_process*: 프로세스 개수 / *num_resource*: resource type 개수.

- 17~22 : resourceUnits 에 각 resource type 별 resource unit 개수 저장. : sum allocated 초기화.
- 24~33 : *allocated* 에 각 프로세스별 allocated matrix 저장 / *sum_allocated* 에 resource type 별 allocated 된 총 resource units 수 저장.
- 35~43 : *requested* 에 각 프로세스별 requested matrix 저장.

2) Int findUnblocked(int reductedProcess[])

- ✓ unblocked process가 존재한다면 해당 프로세스 번호(*findProcess*) 반환 /
 존재하지 않으면 -1 반환.
- ✓ Int reductedProcess[] : 각 process별로 unblocked process라면 1 / blocked process라면 0이 저장된 배열.

```
int findUnblocked(int reductedProcess[]){
47
          int findProcess = -1;
          for(int i=0;i<num_process;i++){</pre>
49
              bool satisfy = true;
              if(reductedProcess[i] == 1) continue;
              for(int j=0;j<num_resource;j++){</pre>
                  if(requested[i][j] > (resourceUnits[j] - sum_allocated[j])){
                      satisfy = false;
                      break;
57
              if(satisfy){
                  findProcess = i;
                  break;
60
         if(findProcess==-1) return -1;
62
         else return findProcess;
```

- 46 ~ 64 : 모든 process에 대한 조사를 위해 반복문 실행.
- 49 : 반복문 실행마다 *satisfy* 변수를 true로 초기화 / *satisfy* = true;
- 50 : 이미 unblocked process는 검사하지 않고 continue문 실행.
- 51 ~ 56 : 모든 resource type에 대한 조사를 위해 반복문 실행.
 - ✓ 52~54 : (request 자원 수 > 총 자원 수 할당된 총 자원)가 존재하는 경우 -> blocked process임 / *satisfy* = false; break;

- ✓ 57~59 : *satisfy* == true 인 경우 -> unblocked process 찾음 / *findProcess* =i; break;
- 62 ~ 63 : unblocked process가 존재한다면 해당 프로세스 번호(*findProcess*) 반환 / 존재하지 않으면 -1 반환.

3) Int main()

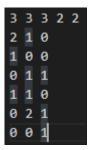
```
v int main(){
         input();
         int reductedProcess[num process];
         for(int i=0;i<num_process;i++) reductedProcess[i]=0;</pre>
         while(1){
70
              int unblockedProcess = findUnblocked(reductedProcess);
71 🗸
              if(unblockedProcess == -1){
                  break:
              }
74 🗸
             else{
75 🗸
                  for(int i=0;i<num_resource;i++){</pre>
76
                      sum_allocated[i] -= allocated[unblockedProcess][i];
78
79
                  reductedProcess[unblockedProcess] = 1;
         int cnt = 0;
         for(int i=0;i<num process;i++){</pre>
84
              if(reductedProcess[i] == 0){
                  cnt = 1;
                  printf("The Deadlocked Process is P%d \n",i+1);
88
         if(cnt == 0) printf("There is no Deadlocked Process!!\n");
90
91
         return 0;
```

- 66 : *input()* 함수 실행.
- 67 ~ 68 : *reductedProcess* 배열 초기화.
- 69 ~ 81 : Graph reduction을 이용한 Deadlock Detection을 위해 while문 실행.
- 70 : *findUnblocked(reductedProcess)* 함수 실행하여 반환 값을 *unblockedProcess*에 저장.
- 71 ~ 73 : unblocked process 찾지 못한 경우 -> while문 break.

- 74 ~ 80 : unblocked process 찾은 경우.
 - ✓ 75~77 : 각 resource type 별로 반복문 실행하여 *sum_allocated* 에 각 resource type 별로 (총 할당된 자원 unblocked process에 할당되었던 자원) 값 저장.
 - ✓ 79: reductedProcess[unblockedProcess] 에 1 저장.
 - ✓ while문 다시 실행.
- 81 ~ 89 : Deadlock Detection 기법 실행 결과 출력.
 - ✓ 83 ~ 88 : Deadlock 존재하는 경우 -> Deadlocked process list 출력.
 - ✓ 89 : Deadlock 존재하지 않는 경우 -> "There is no Deadlocked Process" 출력.

2. 다양한 입력에 대한 실행 결과.

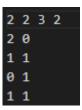
1) Input



Output

```
The Deadlocked Process is P1
The Deadlocked Process is P2
```

2) Input



Output

There is no Deadlocked Process!!

3) Input

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

Output

```
The Deadlocked Process is P1
The Deadlocked Process is P2
```

4) Input

```
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
```

Output

There is no Deadlocked Process!!

5) Input

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

Output

```
The Deadlocked Process is P1
The Deadlocked Process is P2
The Deadlocked Process is P3
The Deadlocked Process is P5
```

6) Input

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

Output

There is no Deadlocked Process!!

7) Input

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

Output

```
The Deadlocked Process is P1
The Deadlocked Process is P2
The Deadlocked Process is P3
```

- 3. 실행환경 및 소스코드 실행 방법.
 - Linux 시스템
 - g++ -o m main.cc
 - ./m

4. 제출물 내 각 파일.

- report.pdf : 보고서

- Main.cc : 최종 소스 코드 파일 (보다 효과적인 자료구조 사용 위해 c++ 언어 사용)