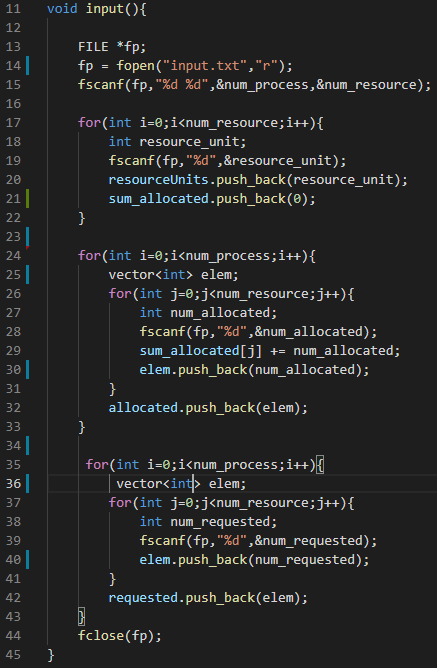
<OS Project 2>

2016312226 박찬순

1. **구현 내용 설명.**

***1) void input()***

* + 파일 입력을 받아 처리.



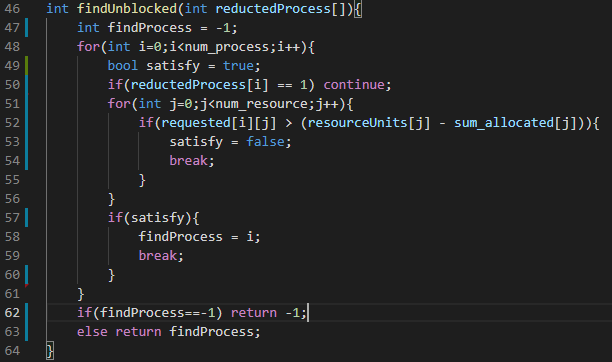
* 13~15 : ***num\_process*** : 프로세스 개수 / ***num\_resource*** : resource type 개수.
* 17~22 **: *resourceUnits*** 에 각 resource type 별 resource unit 개수 저장.

: ***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.

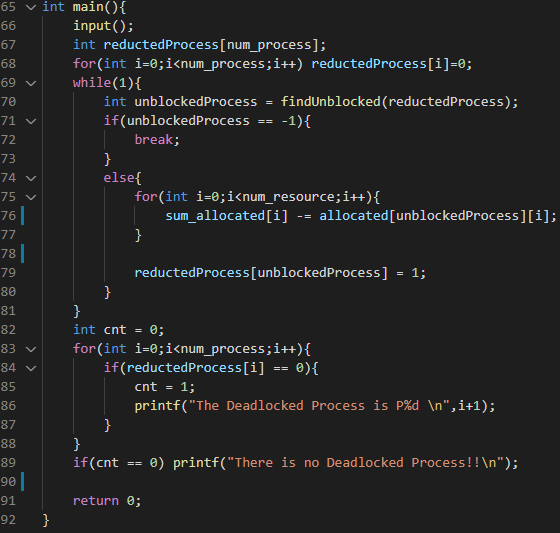
1. ***Int findUnblocked(int reductedProcess[] )***
   * + unblocked process가 존재한다면 해당 프로세스 번호(***findProcess***) 반환 / 존재하지 않으면 -1 반환.

* Int reductedProcess[] : 각 process별로 unblocked process라면 1 / blocked process라면 0이 저장된 배열.



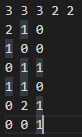
* 46 ~ 64 : 모든 process에 대한 조사를 위해 반복문 실행.
* 49 : 반복문 실행마다 ***satisfy*** 변수를 true로 초기화 / ***satisfy*** = true;
* 50 : 이미 unblocked process는 검사하지 않고 continue문 실행.
* 51 ~ 56 : 모든 resource type에 대한 조사를 위해 반복문 실행.
  + - 52~54 : (request 자원 수 > 총 자원 수 – 할당된 총 자원)가 존재하는 경우 -> blocked process임 / ***satisfy***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

1. ***Int main()***



* 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.

1. **다양한 입력에 대한 실행 결과.**
2. Input



Output



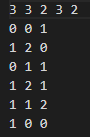
1. Input



Output



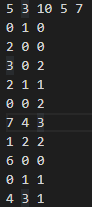
1. Input



Output



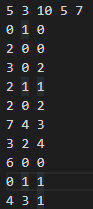
1. Input



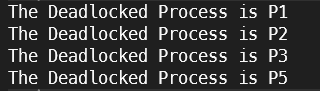
Output



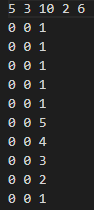
1. Input



Output



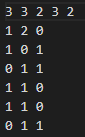
1. Input



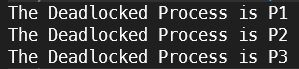
Output



1. Input



Output



1. **실행환경 및 소스코드 실행 방법.**

* Linux 시스템
* g++ -o m main.cc
* ./m

1. **제출물 내 각 파일.**

* report.pdf : 보고서
* Main.cc : 최종 소스 코드 파일 (보다 효과적인 자료구조 사용 위해 c++ 언어 사용)