UNIX 프로그래밍 실습
(2019/11/19)

1. server process는 세 개의 client process들과 데이터를 주고받기 위해 message queue를 만듭니다. 각 client는 message queue를 이용하여, 표준 입력으로 입력된 정수를 server process에게 전송 합니다. server process는 client process로부터 전송된 정수 값에 +8을 한 후, 해당 client에게 다시 보냅니다. client process는 돌려받은 정수 값을 표준 출력으로 출력 합니다. client process는 정수 데이터의 입/출력 작업을 5회 반복 한 후 종료 합니다. 각 client process는 main() 함수의 arguments로 자신의 id를 입력 받습니다.

```
struct q_entry{
        long mtype;
        int data;
};

int main(void){
        int i, qid;
        key_t key;
        struct q_entry msg;

        key=ftok("key", 3);
        // message queue 만들고 open

        for (i=0; i<15; i++) {
                // message 받기
                // mtype 조정
                msg.data+=8;
                // message 보내기
        }

        return 0;
}
```

```
struct q_entry{
        long mtype;
        int data;
};

int main(int argc, char** argv){
        int i, qid, in, id;
        key_t key;
        struct q_entry msg;

        id=atoi(argv[1]);

        key=ftok("key", 3);
        // message queue open

        for (i=0; i<5; i++) {
                scanf("%d", &in);
                // mtype 설정
                // 데이터 복사
                // message 보내기
                // message 받기
                printf("%d₩n", msg.data);
        }

        return 0;
}
```

2. 네 개의 프로세스가 동기화를 하며 자신의 프로세스 id를 5회 출력하는 프로그램을 작성 합니다. 이 프로그램은 main() 함수의 arguments로 동기화에 참여하는 전체 프로세스 중 자신의 출력 순서를 입력 받습니다. 프로그램이 시작되면, 순서대로 자신의 프로세스 id를 출력 합니다. 동기화 작업은 message queue를 사용하여 수행 합니다.