```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>

// 전역 변수 선언
// Just declare balance into a global variable.
int balance = 0; // 계좌 잔액
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER; // 뮤텍스 락
pthread_cond_t newDeposit = PTHREAD_COND_INITIALIZER; // 조건 변수

// DepositTask class will be replaced with a thread named DepositThread
void* depositThread(void* arg) {
    while (1) {
  // Pthread_mutext_lock() instead of using lock()
        pthread_mutex_lock(&lock);

        // 1에서 10 사이의 랜덤 금액을 입금
        int depositAmount = rand() % 10 + 1;
        balance += depositAmount;
        printf("Deposit %d\t\t\t\t\t%d\n", depositAmount, balance);

  // Pthread_cond_signal() instead of using signal()
        pthread_cond_signal(&newDeposit);

        // Pthread_mutext_unlock() instead of using unlock()
        pthread_mutex_unlock(&lock);
```

```
        // 1초 대기
        sleep(1);
    }
    return NULL;
}

// WithdrawTask class will be replaced with a thread named WithdrawThread
void* withdrawThread(void* arg) {
    while (1) {
        // 뮤텍스 잠금
        pthread_mutex_lock(&lock);

        // 출금할 금액을 랜덤으로 결정 (1에서 10 사이)
        int withdrawAmount = rand() % 10 + 1;

        // 계좌 잔액이 충분하지 않으면 조건 변수 대기
        while (balance < withdrawAmount) {
            printf("\t\t\tWait for a deposit\n");
         //Pthread_cond_wait() instead of using await()
            pthread_cond_wait(&newDeposit, &lock);
        }

        // 잔액이 충분할 경우 출금 수행
        balance -= withdrawAmount;
        printf("\t\t\tWithdraw %d\t\t%d\n", withdrawAmount, balance);

        // 뮤텍스 잠금 해제
        pthread_mutex_unlock(&lock);
```

```
    }
    return NULL;
}

int main() {
    // 난수 생성을 위한 시드 초기화
    srand(time(NULL));

    // 스레드 식별자 선언
    pthread_t deposit, withdraw;
    printf("Thread 1\t\tThread 2\t\tBalance\n");

    // 입금 스레드 생성
    pthread_create(&deposit, NULL, depositThread, NULL);
    // 출금 스레드 생성
    pthread_create(&withdraw, NULL, withdrawThread, NULL);

    // 메인 함수가 종료되지 않도록 스레드 조인
    pthread_join(deposit, NULL);
    pthread_join(withdraw, NULL);

    return 0;
}
```