```c
#include <time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>

#define NUMBER_OF_RESOURCES 5
#define NUMBER_OF_CUSTOMERS 5

sem_t mutex;    //lock

int available[NUMBER_OF_RESOURCES];
int maximum[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];

bool need_lt_work(int ith_need[], int work[]);
bool is_safe();
int request_resources(int customer_num, int source_nums[]);
int release_resources(int customer_num, int source_nums[]);
void *customer_thread(int customer_num);
void print_state();
char *to_str(int source_nums[]);

int main(int argc, char const *argv[]){

//available
    if (argc < NUMBER_OF_RESOURCES + 1) {
        printf("Not enough arguments\n");
        return EXIT_FAILURE;
    }
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
        available[i] = atoi(argv[i + 1]);
    }

    srand(time(NULL));
//maximum and need and allocation
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
        for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
            maximum[i][j] = rand()%available[j] + 1;
            need[i][j] = maximum[i][j];
            allocation[i][j] = 0;
        }
    }

    print_state();

    sem_init(&mutex, 0, 1);

//customers
    pthread_t customer_threads[NUMBER_OF_CUSTOMERS];
//create customers
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
        pthread_create(&customer_threads[i], NULL, (void *) customer_thread, (void *)(intptr_t) i);
    }
//join customers
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
        pthread_join(customer_threads[i], NULL);
    }
    return 0;
}

//customer
void *customer_thread(int customer_num) {
    int source_nums[NUMBER_OF_RESOURCES];
    bool req_or_rel;
    int done;
//requesting
    for (int i = 0; i < 30 ; i++){
        req_or_rel = rand() % 2;
// request
        if (req_or_rel == 0) {
            for (int j = 0; j < NUMBER_OF_RESOURCES ; j++){
                source_nums[j] = rand() % (need[customer_num][j] + 1);
            }
            done = request_resources(customer_num, source_nums);
            printf("Customer %d Requests [%s\b] : %s\n", customer_num + 1, to_str(source_nums), done ==
  0 ? "accepted" : "not accepted");
        }
```

```
// release
        else {
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
                source_nums[j] = rand() % (allocation[customer_num][j] + 1);
            }
            done = release_resources(customer_num, source_nums);
            printf("Customer %d Releases [%s\b]\n", customer_num + 1, to_str(source_nums));
        }
    }
}


int request_resources(int customer_num, int source_nums[]) {
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
//(1 & 2)
        if (source_nums[i] > need[customer_num][i] || source_nums[i] > available[i]) {
            return -1;
        }
    }
    sem_wait(&mutex);
//(3)
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
        available[i] -= source_nums[i];
        allocation[customer_num][i] += source_nums[i];
        need[customer_num][i] -= source_nums[i];
    }
    if (is_safe()) {
        sem_post(&mutex);
        return 0;
    }
//undo if not safe
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
        available[i] += source_nums[i];
        allocation[customer_num][i] -= source_nums[i];
        need[customer_num][i] += source_nums[i];
    }
    sem_post(&mutex);
    return -1;
}

int release_resources(int customer_num, int source_nums[]) {
    sem_wait(&mutex);
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
        available[i] += source_nums[i];
        allocation[customer_num][i] -= source_nums[i];
    }
    sem_post(&mutex);
}

bool is_safe() {
    int work[NUMBER_OF_RESOURCES];
//(1)
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
        work[i] = available[i];
    }
    bool finish[NUMBER_OF_CUSTOMERS] = {0};
    int finished = 0;
    bool flag;
    while (finished != NUMBER_OF_CUSTOMERS) {
        flag = false;
        for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
//(2)
            if (!finish[i] && need_lt_work(need[i], work)) {
//(3)
                for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
                    work[j] += allocation[i][j];
                }
                finish[i] = true;
                finished++;
                flag = true;
            }
        }
        if (!flag) {
            return false;
        }
    }
//(4)
    return true;
}
```

```
165 
166 //is the need of one P less than work (for each resource)
167 bool need_lt_work(int ith_need[], int work[]) {
168     for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
169         if (ith_need[i] > work[i]) {
170             return false;
171         }
172     }
173     return true;
174 }
175 
176 //string of requests
177 char *to_str(int source_nums[]) {
178     char *str = malloc(100);
179     char buf[NUMBER_OF_RESOURCES] = {0};
180     for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
181         sprintf(buf, "%d", source_nums[i]);
182         strcat(str, buf);
183         strcat(str, " ");
184     }
185     return str;
186 }
187 
188 //print state
189 void print_state() {
190     printf("Available:\n");
191     for (int i = 0; i < NUMBER_OF_RESOURCES; i++){
192         printf("%d ", available[i]);
193     }
194     printf("\n");
195 
196     printf("Maximum:\n");
197     for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
198         for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
199             printf("%d ", maximum[i][j]);
200         }
201         printf("\n");
202     }
203 
204     printf("Allocation:\n");
205     for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
206         for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
207             printf("%d ", allocation[i][j]);
208         }
209         printf("\n");
210     }
211 
212     printf("Need:\n");
213     for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
214         for (int j = 0; j < NUMBER_OF_RESOURCES; j++){
215             printf("%d ", need[i][j]);
216         }
217         printf("\n");
218     }
219 
220     printf("**********************\n");
221 }
```

```
moujanmirjalili@ubuntu:~/Desktop/lab7_MoujanMirjalili/7$ gcc -pthread -o lab7 lab7.c
moujanmirjalili@ubuntu:~/Desktop/lab7_MoujanMirjalili/7$ ./lab7 1 2 3 4 5
Available:
1 2 3 4 5
Maximum:
1 2 1 3 3
1 1 2 1 4
1 2 1 3 4
1 1 1 1 3
1 1 3 2 5
Allocation:
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Need:
1 2 1 3 3
1 1 2 1 4
1 2 1 3 4
1 1 1 1 3
1 1 3 2 5
**********************
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Requests [0 1 0 0 0] : accepted
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 1 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 2 Requests [0 1 2 1 4] : not accepted
Customer 1 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 1 Requests [1 1 0 2 2] : accepted
Customer 1 Requests [0 0 1 1 0] : accepted
Customer 1 Releases [1 0 0 3 1]
Customer 2 Releases [0 0 0 0 0]
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 1] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [1 1 1 0 3] : not accepted
Customer 2 Releases [0 0 0 0 0]
Customer 2 Releases [0 0 0 0 0]
Customer 2 Requests [0 0 2 0 0] : not accepted
Customer 2 Releases [0 0 0 0 0]
Customer 4 Requests [0 1 1 0 2] : accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [0 2 1 2 4] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [0 2 0 3 3] : not accepted
Customer 3 Requests [0 2 0 1 2] : not accepted
Customer 3 Requests [1 0 0 2 0] : not accepted
Customer 4 Releases [0 0 1 0 2]
Customer 4 Requests [1 0 0 1 1] : accepted
Customer 4 Releases [1 1 0 0 1]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Releases [0 0 0 1 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 4 Requests [0 0 0 0 0] : accepted
```

```
Customer 4 Releases [0 0 0 0 0]
Customer 4 Releases [0 0 0 0 0]
Customer 4 Releases [0 0 0 0 0]
Customer 4 Releases [0 0 0 0 0]
Customer 4 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Releases [0 0 0 0 2]
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Releases [0 1 1 0 0]
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Requests [0 0 0 0 0] : accepted
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 1 Releases [0 0 0 0 0]
Customer 2 Releases [0 0 0 0 0]
Customer 2 Requests [1 0 2 1 2] : accepted
Customer 2 Requests [0 1 0 0 2] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Releases [1 0 1 1 0]
Customer 2 Releases [0 0 0 0 2]
Customer 2 Releases [0 0 0 0 2]
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Releases [0 0 1 0 0]
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Releases [0 1 0 0 0]
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Releases [0 0 0 0 0]
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Releases [0 0 0 0 0]
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 2 Requests [0 0 0 0 0] : accepted
Customer 5 Releases [0 0 0 0 0]
Customer 5 Releases [0 0 0 0 0]
Customer 5 Releases [0 0 0 0 0]
Customer 3 Requests [1 2 1 1 1] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [0 1 1 0 0] : not accepted
Customer 3 Requests [1 0 1 1 3] : not accepted
Customer 3 Requests [1 2 1 3 0] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [1 0 0 2 0] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [1 2 0 2 4] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 3 Releases [0 0 0 0 0]
Customer 3 Requests [0 0 0 0 3] : not accepted
Customer 3 Requests [1 2 0 0 0] : not accepted
Customer 3 Requests [0 2 1 2 0] : not accepted
Customer 3 Requests [1 2 0 1 3] : not accepted
Customer 3 Requests [0 0 1 2 2] : not accepted
Customer 3 Requests [0 0 1 3 1] : not accepted
Customer 3 Releases [0 0 0 0 0]
Customer 5 Requests [1 0 1 1 4] : accepted
Customer 5 Releases [0 0 1 1 0]
Customer 5 Requests [0 0 0 0 1] : accepted
Customer 5 Requests [0 1 1 0 0] : accepted
Customer 5 Requests [0 0 0 1 0] : accepted
Customer 5 Releases [1 1 1 1 4]
Customer 5 Releases [0 0 0 0 0]
Customer 5 Requests [0 0 1 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Releases [0 0 0 0 0]
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Releases [0 0 0 0 1]
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Releases [0 0 0 0 0]
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Releases [0 0 1 0 0]
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
Customer 5 Requests [0 0 0 0 0] : accepted
```