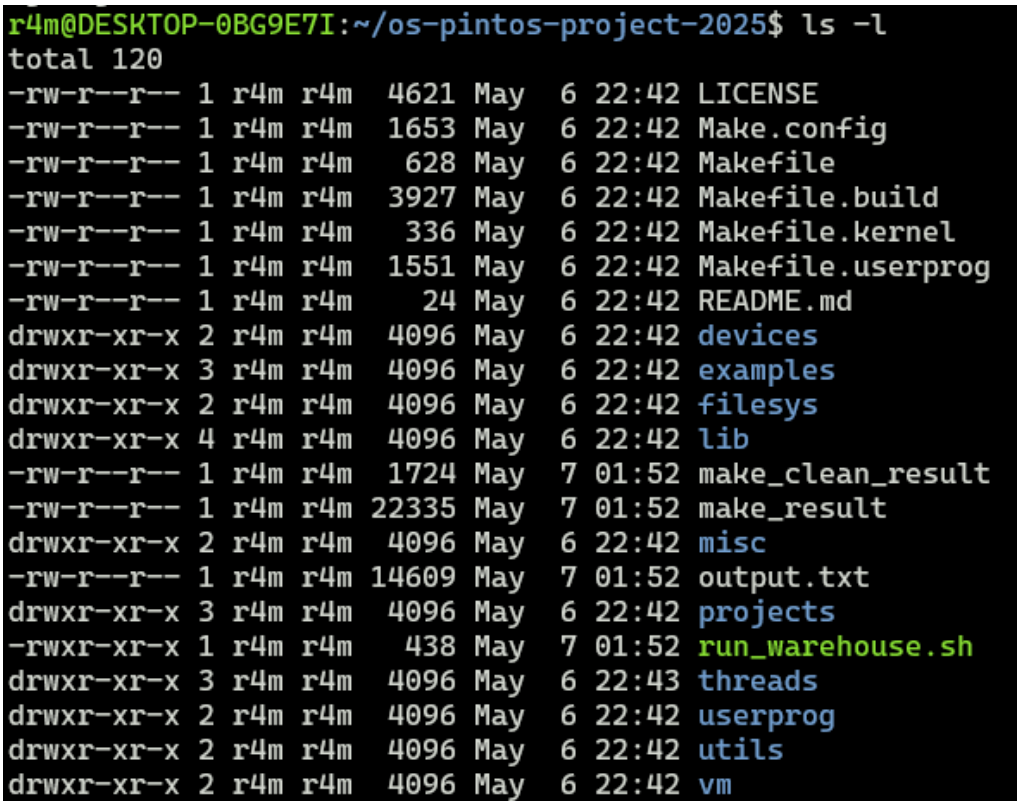
스레드 관리:
자동화 물류 센터 시뮬레이터 개발

[2025 Operating System Project#1]

- 2025.05.07 -

학번: 20214234
이름: 김하람

1. 프로젝트 요구사항 #0
   - Pintos 환경에서 빌드 및 실행을 위해 wsl 환경에서 프로젝트 폴더를 구축하였고, run_warehouse.sh 쉘 스크립트를 제작하여 프로젝트 빌드 및 실행을 원활하게 하였음. 에러가 발생할 경우를 대비하여 make_result 파일과 make_clean_result 파일을 생성하게 하여 디버깅을 하였음.
   - 프로그램의 실행 결과는 output.txt에 저장함.
   - ./run_warehouse.sh {robot_count} {물건 번호와 하역 장소 쌍의 리스트} 명령으로 프로그램을 쉽게 실행할 수 있음



2. 프로젝트 요구사항 #1
   a. 시뮬레이터 실행 파라미터 입력 값을 파싱하는 동작은 parse_parametes 함수에서 처리함. 첫 번째 인자는 robot_count 전역 변수에 저장, 두 번째 인자는 strtok_r 함수를 주로 이용하여 colon 문자를 기준으로 파싱하여 robot_tasks 배열에 저장함.

```c
// Parse command line arguments and set variables
void parse_parameters(char **argv) {

        // [1] Set number of robots
        robot_count = atoi(argv[1]);

        // [2] Set robots tasks
        robot_tasks = malloc(sizeof(char *) * robot_count);

        char buffer[256];
        strlcpy(buffer, argv[2], sizeof(buffer));
        buffer[sizeof(buffer) - 1] = '\0';

        char *saveptr;
        char *token = strtok_r(buffer, ":", &saveptr);
        int index = 0;

        while (token && index < robot_count) {
                robot_tasks[index] = malloc(3);
                strlcpy(robot_tasks[index], token, 3);
                robot_tasks[index][2] = '\0';
                token = strtok_r(NULL, ":", &saveptr);
                index++;
        }
}
```

   b. 파싱된 파라미터 값을 기준으로 중앙 관제 노드 스레드(cnn_thread)는 run_automated_warehouse 함수 내에서 생성하였음.

```c
// entry point of simulator
void run_automated_warehouse(char **argv)
{
        init_automated_warehouse(argv); // do not remove this

        printf("\n\n====================================\n");
        printf("*** Automated Warehouse Simulator ***\n");
        printf("***        20214234 김하람        ***\n");
        printf("====================================\n\n\n");

        // [1] Parse parameters
        parse_parameters(argv);

        // [2] Create Central Control Node threads
        thread_create("CNT", 0, &cnn_thread, NULL);
}
```

물류 로봇들에 대한 스레드(robot_thread)는 중앙 관제 노드 내에서 robot 구조체 배열 생성(init_robots) 이후 init_robots_threads 함수를 통해 스레드 생성함.

```c
// Thread for Central Control Node
void cnn_thread() {

        // [1] Create and Set robots
        init_robots();

        // [2] Create robots threads
        init_robots_threads();
```

```c
// Initialize robot threads
void init_robots_threads() {
        for (int i = 0; i < robot_count; i++) {
                struct robot *r = &robots[i];
                thread_create(r->name, 0, &robot_thread, r);
        }
}
```

3. 프로젝트 요구사항 #2
   - 스레드 간 데이터 전송을 위한 Block, Unblock 함수 구현

   - 이전 인터럽트 값을 old_level 변수에 저장 후 block_threads 리스트에 push 수행. 이후 thread_block 함수를 호출하여 스레드를 block 상태로 변경. 스레드를 block 후에 intr_set_level(old_level) 함수를 호출하여 이전 인터럽트 값으로 설정하여 선점을 방지함.

```
void block_thread(){
    enum intr_level old_level = intr_disable ();
    list_push_back(&blocked_threads, &thread_current()->elem);
    thread_block();
    intr_set_level(old_level);
}
```

   - block_thread와 마찬가지로 이전 인터럽트 값을 old_level 변수에 저장 후 block_threads 리스트에 있는 모든 값들을 꺼내어 unblock 상태로 변경. 모든 스레드에 대해 unblock을 수행한 후에 이전 인터럽트 값으로 설정함.

```
void unblock_threads(){
    enum intr_level old_level = intr_disable ();
    while(!list_empty(&blocked_threads)){
        struct list_elem *e = list_pop_front(&blocked_threads);
        struct thread *t = list_entry(e, struct thread, elem);
        thread_unblock(t);
    }
    intr_set_level(old_level);
}
```

4. 프로젝트 요구사항 #3
   - block_thread 및 unblock_threads를 통한 스레드 간 데이터 전달 시스템 구현

a. message 구조체를 이용하여 스레드 간 데이터 전달 내용을 저장함. 각 변수는 구현에 필요한 역할로 해석하여 사용하였으며, is_stopped 변수를 새로 추가함.

in aw_message.h

변수	역할
로봇 -> 중앙 관제 노드	
int row	현재 로봇이 위치한 row 값
int col	현재 로봇이 위치한 col 값
int current_payload	현재 로봇이 물건을 들고 있는지를 나타내는 값 (0: 들고 있지 않음 / 1: 들고 있음)
int required_payload	현재 로봇이 적재와 하역해야 하는 물건이 존재하는지를 나타내는 값 (0: 존재하지 않음 / 1: 존재함)
int is_stopped	현재 로봇이 멈춰 있는지를 나타내는 값 (0: 멈춰 있지 않음 / 1: 멈춰 있음)
중앙 관제 노드 -> 로봇	
int cmd	로봇이 다음으로 수행해야 할 명령
int target_row	로봇이 다음으로 움직여야 할 row 값
int target_col	로봇이 다음으로 움직여야 할 col 값

```
struct message {
    //
    // To central control node
    //
    /** current row of robot */
    int row;
    /** current column of robot */
    int col;
    /** current payload of robot */
    int current_payload;
    /** required paylod of robot */
    int required_payload;
    /** check if the robot is stopped */
    int is_stopped;

    //
    // To robots
    //
    /** next command for robot */
    int cmd;
    int target_row;
    int target_col;
};
```

b. 중앙 관제 노드가 각 로봇에게 전달해야 할 메시지를 전달하는 메시지 함 message_box는 boxes_from_central_node 구조체 배열에 인덱스로 구분하여 저장하게 함.

각 로봇이 중앙 관제 노드에게 전달해야 할 메시지를 전달하는 메시지 함 message_box는 boxes_from_robots 구조체 배열에 인덱스로 구분하여 저장하게 함.

각 message_box 구조체 배열의 동적 할당은 init_message_boexs 함수에서 수행함.

```
// Initialize the message boxes
void init_message_boxes() {
        boxes_from_central_control_node = malloc(sizeof(struct message_box) * robot_count);
        boxes_from_robots = malloc(sizeof(struct message_box) * robot_count);
        for (int i = 0; i < robot_count; i++){
                boxes_from_central_control_node[i].dirtyBit = 0;
                boxes_from_robots[i].dirtyBit = 0;
        }
}
```

message_box는 수신자가 메시지를 읽었는지를 나타내기 위해 dirtyBit를 사용하였고, 0일 때 읽었음을 나타내고 1일 때 읽지 않았음을 나타냄.

```
/**
 * Simple message box which can receive only one message from sender
*/
struct message_box {
    /** check if the message was written by others */
    int dirtyBit;
    /** stored message */
    struct message msg;
};

/** message boxes from central control node to each robot */
extern struct message_box* boxes_from_central_control_node;
/** message boxes from robots to central control node */
extern struct message_box* boxes_from_robots;
```

c. 물류 로봇은 while-loop 내에서 [1]맨 처음 block 이후 unblock이 되면 [2]중앙 관제 노드에게 데이터를 전달한 후에 block 됨.

```
// Main loop
while(1){

        // [1] Wait until the robot is unblocked
        block_thread();


        // [2] Check message box
        if (boxes_from_central_control_node[r->index].dirtyBit) {
                struct message msg = boxes_from_central_control_node[r->index].msg;

                // [2-1] Process the message
```

[9-1]중앙 관제 노드는 모든 물류 로봇에게 데이터를 전달 받은 후에 데이터를 기반으로 각 로봇에게 명령을 전달한 다음, [9-5] block 상태의 로봇 스레드들을 unblock 하여 주었음.

```
// [9] Main loop
while (!check_all_robots_done()) {

        // [9-1] Process messages from robots
        for (int i = 0; i < robot_count; i++) {
                if (boxes_from_robots[i].dirtyBit) {
                        struct message msg = boxes_from_robots[i].msg;

                        // Process the message
                        current_robot_row[i] = msg.row;
                        current_robot_col[i] = msg.col;
                        current_robot_payload[i] = msg.current_payload;
                        current_robot_required_payload[i] = msg.required_payload;
                        current_is_stopped[i] = msg.is_stopped;

                        // Reset the dirty bit
                        boxes_from_robots[i].dirtyBit = 0;
                }
        }
```

```
                        // Assign next move to the robot
                        assign_next_move(r, 1, steps[i]);
                        current_is_stopped[i] = 0;
                        steps[i]++;
                }

        }

        // [9-5] Unblock all threads
        unblock_threads();

        // [9-6] Increase step
        increase_step();

        // [9-7] Timer sleep for 1 second
        timer_msleep(1000);
}
```

5. 프로젝트 요구사항 #4
   a. 우선, 필요한 역할에 따라 robot 구조체의 멤버 변수를 추가함.

변수	역할
const char* name	로봇의 이름 R1부터 시작
int index	로봇의 인덱스 값
int row	로봇이 위치한 현재 row 값
int col	로봇이 위치한 현재 col 값
int current_payload	로봇이
int current_payload	현재 로봇이 물건을 들고 있는지를 나타내는 값 (0: 들고 있지 않음 / 1: 들고 있음)
int required_payload	현재 로봇이 적재와 하역해야 하는 물건이 존재하는지를 나타내는 값 (0: 존재하지 않음 / 1: 존재함)
char load_location	적재 물건이 존재하는 장소
char unload_location	하역장 장소
int load_location_row	적재 물건이 존재하는 장소의 row 값
int load_location_col	적재 물건이 존재하는 장소의 col 값
int unload_location_row	하역장 장소의 row 값
int unload_location_col	하역장 장소의 col 값
int priority	로봇 간의 우선 순위를 나타내는 값 낮은 숫자일수록 큰 우선 순위 인덱스를 바탕으로 부여함
int is_stopped	현재 로봇이 멈춰 있는지를 나타내는 값 (0: 멈춰 있지 않음 / 1: 멈춰 있음)

```c
/**
 * A Structure representing robot
 */
struct robot {
    const char* name;
    int index;
    int row;
    int col;
    int current_payload;
    int required_payload;

    char load_location;
    char unload_location;

    int load_location_row;
    int load_location_col;
    int unload_location_row;
    int unload_location_col;

    int priority; // Priority of the robot (lower number means higher priority)

    int is_stopped; // 0: moving, 1: stopped
};
```

b. 중앙 관제 노드는 물류 로봇 배열을 동적으로 할당해주고, setRobot 함수를 통해 초기 위치인 W 위치에 위치시킴.

setRobot 함수의 마지막 인자는 추후에 로봇끼리 충돌 검사를 수행할 때 사용할 로봇의 우선 순위를 로봇의 인덱스 값으로 설정하였음.

```c
// Initialize robots
void init_robots() {
        robots = malloc(sizeof(struct robot) * robot_count);
        robot_names = malloc(sizeof(char *) * robot_count);
        for (int i = 0; i < robot_count; i++){
                robot_names[i] = malloc(sizeof(char) * 3);
                snprintf(robot_names[i], sizeof(robot_names), "R%d", i + 1);

                // Set robot and place robot to W
                setRobot(&robots[i], robot_names[i], i, 6, 5, i);
        }
}
```

앞서 파싱 하였던 파라미터 값을 바탕으로 물류 로봇이 요구하는 물건 적재 지역과 하역 장소를 robot의 멤버 변수인 load_location과 unload_location을 참조하여 해당 장소의 row, col 값을 저장함.

```c
// Set destination to load location
void set_destination_to_load_location() {

        for (int i = 0; i < robot_count; i++) {
                struct robot *r = &robots[i];
                int flag = 0;

                for (int j = 0; j < MAP_HEIGHT; j++) {
                        for (int k = 0; k < MAP_WIDTH; k++) {
                                if (flag) break;
                                if (map_draw_default[j][k] == r->load_location) {
                                        r->load_location_row = j;
                                        r->load_location_col = k;

                                        flag = 1;
                                        break;
                                }
                        }
                }
        }
}
```

```c
// Set destination to unload location
void set_destination_to_unload_location() {

        for (int i = 0; i < robot_count; i++) {
                struct robot *r = &robots[i];
                int flag = 0;

                for (int j = 0; j < MAP_HEIGHT; j++) {
                        for (int k = 0; k < MAP_WIDTH; k++) {
                                if (flag) break;

                                if (map_draw_default[j][k] == r->unload_location) {
                                        r->unload_location_row = j;
                                        r->unload_location_col = k;

                                        flag = 1;
                                        break;
                                }
                        }
                }
        }
}
```

각 물류 로봇은 boxes_from_central_control_node로부터 전달받은 명령(msg.cmd)을 바탕으로 switch-case문 내에서 {이동, 적재, 하역, 대기}의 4가지 명령 중 한 가지를 수행함.

```
// [2] Check message box
if (boxes_from_central_control_node[r->index].dirtyBit) {
        struct message msg = boxes_from_central_control_node[r->index].msg;

        // [2-1] Process the message
        switch (msg.cmd) {
                case CMD_MOVE:
                r->row = msg.target_row;
                r->col = msg.target_col;
                r->is_stopped = 0;
                break;

                case CMD_LOAD:
                r->current_payload = 1;
                r->is_stopped = 1;
                break;

                case CMD_UNLOAD:
                r->current_payload = 0;
                r->required_payload = 0;
                r->is_stopped = 1;
                break;

                case CMD_WAIT:
                r->is_stopped = 1;
                break;
        }
```

이후에는 boxes_from_central_control_node의 dirtyBit를 0으로 설정하여 메시지를 읽었음을 표시하고, boxes_from_robots를 통해 상태 정보를 전달 후 dirtyBit를 1로 설정하여 메세지를 전달하였음을 나타냄.

```
// [2-2] Reset dirty bit
boxes_from_central_control_node[r->index].dirtyBit = 0;

// [2-3] Process the message
boxes_from_robots[r->index].msg.row = r->row;
boxes_from_robots[r->index].msg.col = r->col;
boxes_from_robots[r->index].msg.current_payload = r->current_payload;
boxes_from_robots[r->index].msg.required_payload = r->required_payload;
boxes_from_robots[r->index].msg.is_stopped = r->is_stopped;

// [2-4] Set dirty bit
boxes_from_robots[r->index].dirtyBit = 1;
```

c. 중앙 관제 노드는 물류 로봇이 이동하기 위한 조건을 고려하여 모든 물류 로봇들이 각자의 적재 장소로 이동하여 물건 적재 이후 하역 장소로 이동하여 물건을 하역 하도록 제어함.

물류 로봇이 이동하기 위해서는 요구 사항을 고려하여 (시작점 -> 적재 장소)와 (적재 장소 -> 하역 장소)로의 이동 경로 탐색을 수행한 후에 경로에 따라 로봇이 움직이게 함.

이동 경로 탐색은 너비 우선 탐색(Breadth First Search) 알고리즘을 이용하여 최단 경로를 계산하였음. find_shortest_path_by_bfs 함수를 통하여 각 로봇이 할당 받은 적재 장소와 하역 장소에 따라 bfs 함수를 통하여 탐색한 최단 경로를 전역으로 선언된 shortest_path_by_bfs 구조체 배열에 저장하였음. bfs는 Queue 자료 구조를 활용하여 구현함.

변수	역할
shortest_path_by_bfs[i]	i번째 인덱스를 가진 로봇의 최단 경로를 저장 (i < robot_count)
shortest_path_by_bfs[i][j]	j 값에 따라 경로의 타입을 달리 저장 j=0 : (시작점 -> 적재 장소) 경로 j=1 : (적재 장소 -> 하역 장소) 경로
shortest_path_by_bfs[i][j][k]	각 경로에 따라 순서대로 location 값이 저장 (k < MAP_HEIGHT * MAP_WIDTH)

```
Location*** shortest_path_by_bfs;
```

```
// Start finding shortest path by BFS
void find_shortest_path_by_bfs() {

        // [1] Initialize the shortest path array
        shortest_path_by_bfs = malloc(sizeof(Location**) * robot_count);
        for (int i = 0; i < robot_count; i++) {
            shortest_path_by_bfs[i] = malloc(sizeof(Location*) * 2);

            for (int j = 0; j < 2; j++) {
                shortest_path_by_bfs[i][j] = malloc(sizeof(Location) * MAX_PATH_LENGTH);

                for (int k = 0; k < MAX_PATH_LENGTH; k++) {
                    shortest_path_by_bfs[i][j][k].row = -1;
                    shortest_path_by_bfs[i][j][k].col = -1;
                }
            }
        }

        // [2] Find the shortest path for each robot
        for (int i = 0; i < robot_count; i++) {
                struct robot *r = &robots[i];

                Location start = {ROW_5, COL_5};
                Location load_location = {r->load_location_row, r->load_location_col};
                Location unload_location = {r->unload_location_row, r->unload_location_col};

                // [2-1] Find the path from start to load location and from load location to unload location
                int is_found_path_1 = bfs(start, load_location, shortest_path_by_bfs[i][0], r);
                int is_found_path_2 = bfs(load_location, unload_location, shortest_path_by_bfs[i][1], r);

                if (is_found_path_1 == -1 || is_found_path_2 == -1) {
                        printf("*** Error: No path found for robot %s to load/unload location ***\n", r->name);
                        printf("*** Shutting down simulator .. ***\n");
                        shutdown_power_off();
                }

        }
}
```

shortest_path_by_bfs 구조체 배열이 저장되면, 물류 로봇이 움직여야 할 경우에 assign_next_move 함수를 호출하여 path_type(0: 시작 장소 -> 적재 장소 / 1: 적재 장소 -> 하역 장소)에 따라 다음 경로로 로봇이 이동하도록 row 값과 col 값을 전달해줌. 이때, step 배열을 사용하여 각 로봇별로 경로를 따라 몇 번째 위치까지 이동하였는지를 추적함.

```
// Assign next move to the robot
void assign_next_move(struct robot *r, int path_type, int step) {
        Location next_location = shortest_path_by_bfs[r->index][path_type][step];
        send_command_to_robot(r, CMD_MOVE, next_location.row, next_location.col);
}
```

물류 로봇이 [9-4-c]적재 장소로 이동해야 할 경우와 [9-4-d]하역 장소로 이동해야 할 경우는 로봇 구조체 변수가 가지고 있는 current_payload 값과 required_payload 값에 따라 올바른 경로로 움직이도록 제어하였음.

current_payload	required_payload	상태
0	0	모든 작업 종료
0	1	적재 장소로 이동
1	1	하역 장소로 이동

물류 로봇은 next_location_row 값과 next_location_col 값이 모두 -1일 경우에는 적재 장소 혹은 하역 장소에 도착한 상태이므로 적재 및 하역 작업을 수행하도록 명령함.

물류 로봇이 다음 경로로 움직이기 전에, check_collision 함수를 통해 다른 로봇과 충돌하지 않는지를 확인함. check_collision 함수는 다음의 일련 과정을 통해 충돌 검사를 수행함.

i. 현재 이동하려는 로봇이 적재 혹은 하역 중이라면 다음 이동 위치는 현재 위치로 지정하고, 적재 혹은 하역 중이 아니라면 shortest_path_by_bfs 배열에서 다음 경로를 꺼내어 지정

ii. 다른 모든 로봇에 대해서 다음 이동 위치를 계산. 이때, 하역 장소에는 여러 로봇이 존재할 수 있기에 해당 로봇의 current_payload 값과 required_payload 값이 모두 0이면 충돌 없음 반환.

    다른 로봇이 어떠한 이유이든지 멈춰 있는 상태거나 다른 로봇이 적재 혹은 하역 중일 때에는 다음 이동 위치는 현재 위치로 지정. 그 경우가 아니라면 shortest_path_by_bfs 배열에서 다음 경로를 꺼내어 지정.

iii. 앞서 계산한 현재 이동하려는 로봇과 다른 모든 로봇과의 다음 이동 위치가 동일할 때, 현재 이동 로봇의 우선 순위가 더 작다면 충돌 발생 반환. 그 경우가 아니라면 충돌 없음 반환.

```
// [9-4-c] When robot is moving to load location
if (received_robot_current_payload[i] == 0) {
        Location next_location = shortest_path_by_bfs[r->index][0][steps[i]];

        // When robot is at load location, then load the payload
        if (next_location.row == -1 && next_location.col == -1) {
                send_command_to_robot(r, CMD_LOAD, r->row, r->col);
                steps[i] = 1; // set to 1 to move to unload location
                continue;
        }

        // When collision occurs, then wait
        if(check_collision(r, temp_steps, received_is_stopped)) {
                send_command_to_robot(r, CMD_WAIT, r->row, r->col);
                received_is_stopped[i] = 1;
                continue;
        }

        // Assign next move to the robot
        assign_next_move(r, 0, steps[i]);
        received_is_stopped[i] = 0;
        steps[i]++;

}
```

```
// [9-4-d] When robot is moving to unload location
else if (received_robot_current_payload[i] == 1) {
        Location next_location = shortest_path_by_bfs[r->index][1][steps[i]];

        // When robot is at unload location, then unload the payload
        if (next_location.row == -1 && next_location.col == -1) {
                send_command_to_robot(r, CMD_UNLOAD, r->row, r->col);
                continue;
        }

        // When collision occurs, then wait
        if(check_collision(r, temp_steps, received_is_stopped)) {
                send_command_to_robot(r, CMD_WAIT, r->row, r->col);
                received_is_stopped[i] = 1;
                continue;
        }

        // Assign next move to the robot
        assign_next_move(r, 1, steps[i]);
        received_is_stopped[i] = 0;
        steps[i]++;
}
```

상태 정보를 기반으로 각 물류 로봇에게 명령을 지시한 후에 모든 물류 로봇 스레드를 unblock하여 작업을 수행하도록 하고, increase_step() 호출. 이후 중앙 관제 노드 스레드가 1초 동안 sleep하는 타이머를 작동하여 시뮬레이터처럼 작동하도록 함.

```
// [9-5] Unblock all threads
unblock_threads();

// [9-6] Increase step
increase_step();

// [9-7] Timer sleep for 1 second
timer_msleep(1000);
```

[10]모든 로봇이 작업 수행을 마치면 while loop를 빠져나와 시뮬레이션을 종료함.

```
// [10] All robots are done for the tasks so shutdown the simulator
printf("\n\n====================================\n");
printf("*** All robots are done for tasks ***\n");
printf("***   Shutting Down Simulator ..  ***\n");
printf("====================================\n\n\n");
shutdown_power_off();
```

6. 프로젝트 요구사항 #5
    - 물류 로봇 스레드는 unblock 되기 전까지 대기하고 있고, 중앙 관제 노드는 모든 물류 로봇 스레드에게 작업을 지정해주고, unblock_threads 함수를 호출함으로써 모든 물류 로봇 스레드를 unblock 상태로 지정함. 이후 물류 로봇 스레드는 지정 받은 작업을 동시에 수행함. 중앙 관제 노드는 동시에 물류 로봇 스레드를 제어하는 효과를 볼 수 있음. (앞선 요구 사항에서 해당 동작 과정을 설명하였음)

7. 프로젝트 요구사항 #6

a. Trouble Shooting

과제를 진행하며 마주했던 문제는 크게 2가지였음. 첫 번째는 물류 로봇들이 무작위로 움직이며 적재 장소와 하역 장소에 도달하는 것이 아니라 효율적인 알고리즘으로 이동하도록 해야 하는 문제, 두 번째는 물류 로봇들이 동시에 제어되면서 같은 위치에 이동하려고 하며 발생하는 충돌 문제였음.

i. 물류 로봇의 최단 경로 탐색

물류 로봇이 시작 지점부터 적재 장소까지와 적재 장소부터 하역 장소까지 이동하는 2가지 최단 경로가 필요함. 그리고 물류 로봇이 현재 적재 장소로 이동하는지 하역 장소로 이동하는지를 판단하고 올바른 경로로 중앙 관제 노드가 각 물류 로봇들에게 올바른 명령을 내려야 함.

find_shortest_path_by_bfs 함수에서 bfs 함수를 호출함으로써 각 물류 로봇별 시작 지점부터 적재 장소까지 경로를 shortest_path_by_bfs[i][0] 구조체 배열에, 적재 장소부터 하역 장소까지 경로를 shortest_path_by_bfs[i][1] 구조체 배열에 저장하도록 하였음. 이 구조체 배열은 [9-4-c][9-4-d]이후 물류 로봇이 어떤 경로를 이동해야 하는지를 조건식을 통해 구분함.

```
// [9-4-c] When robot is moving to load location
if (received_robot_current_payload[i] == 0) {
        Location next_location = shortest_path_by_bfs[r->index][0][steps[i]];
```

```
// [9-4-d] When robot is moving to unload location
else if (received_robot_current_payload[i] == 1) {
        Location next_location = shortest_path_by_bfs[r->index][1][steps[i]];
```

bfs로 경로를 탐색할 때에 Queue 자료구조를 활용하여 탐색한 각 위치를 저장해두고 한 경로의 탐색을 마칠 때마다 queue에서 위치 정보를 꺼내어 재 탐색하는 기본적인 Breadth First Search 방식을 이용함. 이때, 이동 가능한 경로를 탐색할 때 아래와 같은 조건식이 True가 되는 경우에만 이동 가능 경로로 추가하였음.

Condition 1. next_row >= 0 && next_row < MAP_HEIGHT
: 맵의 높이(row) 값 내에 위치 확인
Condition 2. next_col >= 0 && next_col < MAP_WIDTH
: 맵의 너비(col) 값 내에 위치 확인
Condition 3. map_draw_default[next_row][next_col] != 'X'
: 이동 불가 장소('X')인지 확인
Condition 4. (map_draw_default[next_row][next_col] == ' '
|| map_draw_default[next_row][next_col] == 'S'
|| map_draw_default[next_row][next_col] == r->load_location

|| map_draw_default[next_row][next_col] == r->unload_location)

: 이동 가능 장소(' ')인지 or 시작 장소 표시('S')인지 or 적재 장소('1' ~'7')나 하역 장소('A'~ 'C')인지 확인

Condition 5. !visited[next_row][next_col]

: 이미 방문한 곳이 아닌지 확인 (bfs 알고리즘에서 사용)

ii. 물류 로봇 간의 충돌 검사

물류 로봇이 다음으로 이동하기 전에 다른 로봇이 다음으로 이동할 경로와 동일한지를 검사하면 로봇들이 같은 장소에 위치하지 않게 만들 수 있음.

check_collition 함수를 통해 함수의 반환 값이 0이면 충돌 없음, 1이면 충돌 있음을 나타내어 중앙 관제 노드가 각 물류 로봇들에게 이동 명령을 전달하기 전에 해당 함수를 호출하여 충돌 발생 시 해당 로봇은 대기 상태(CMD_WAIT) 명령을 전달하고, 충돌이 발생하지 않을 시 해당 로봇은 다음 경로로 이동하는 명령을 전달함.

```
// When collision occurs, then wait
if(check_collision(r, temp_steps, received_is_stopped)) {
        send_command_to_robot(r, CMD_WAIT, r->row, r->col);
        received_is_stopped[i] = 1;
        continue;
}

// Assign next move to the robot
assign_next_move(r, 0, steps[i]);
received_is_stopped[i] = 0;
steps[i]++;
```

check_collision 함수는 총 3가지 인자를 받음.

인자	역할
struct robot *r	현재 이동하고자 하는 로봇의 구조체 포인터
int *steps	모든 로봇들이 shortest_path_by_bfs 배열을 참조하여 이동해야 할 다음 위치의 인덱스(step)을 저장하는 배열
int *is_stopped	모든 로봇들이 현재 이동 정지 상태인지를 저장하는 배열. 다른 로봇이 정지 상태일 때 해당 위치로 이동하지 않게 하기 위해 도입한 배열.

함수 내부에서는 먼저, [1]현재 이동하고자 하는 로봇이 다음으로 이동할 위치 정보를 shortest_path_by_bfs 배열을 참조하여 next_row와 next_col 변수에 저장

함. 만약 두 값 모두 -1 값을 가진다면, 이는 적재 장소 혹은 하역 장소에 도달하여 적재 혹은 하역 작업을 하고 있다는 것을 의미하기에 다음 이동 위치를 현재 위치로 저장함.

```
// [1] Assign next location for current robot
int next_row = shortest_path_by_bfs[r->index][path_type][steps[r->index]].row;
int next_col = shortest_path_by_bfs[r->index][path_type][steps[r->index]].col;
if (next_row == -1 && next_col == -1) {
        next_row = r->row;
        next_col = r->col;
}
```

그 다음으로 [2]for-loop를 통해 다른 모든 로봇의 다음 이동 위치 정보를 other_robot_row, other_robot_col 변수에 저장함. 이때, [2-1]로봇의 current_payload와 required_payload 값을 참조하여 두 값이 모두 0이면 로봇에게 할당 받은 물건 적재 하역 작업을 종료한 것이기에 충돌하지 않음. 0 값을 반환.

```
// [2-1] Skip when other robot is done for the task
if (other_robot->current_payload == 0 && other_robot->required_payload == 0) {
        return 0;
}
```

그 다음으로 인자로 전달받은 is_stopped 배열을 참조하여 다른 로봇이 정지 중인지를 확인함. 로봇이 정지하는 경우는 이전 검사에서 충돌이 발생하여 대기 중이거나, 적재 및 하역 작업을 하고 진행 중이기에 이동 정지 중인 경우임. [2-2] 다른 로봇이 정지 상태인 경우, 다음 이동 위치를 현재 위치로 할당함. [2-3]다른 로봇이 정지 상태가 아닌 경우, 다음 이동 위치를 shortest_path_by_bfs 배열을 참조하여 다음 이동 경로를 할당함. 이때, [2-4]다른 로봇이 적재 혹은 하역 장소에 도달했을 경우, 적재 혹은 하역 작업을 하기 위해 다음 이동 경로를 현재 위치로 할당함.

```
// [2-2] If other robot is stopped, assign its current location
if (is_stopped[i] == 1) {
        other_robot_next_row = other_robot->row;
        other_robot_next_col = other_robot->col;
}
// [2-3] If other robot is moving, assign its next location
else {
        int other_path_type = 0;
        if (other_robot->current_payload == 1) other_path_type = 1;

        other_robot_next_row = shortest_path_by_bfs[i][other_path_type][steps[i]].row;
        other_robot_next_col = shortest_path_by_bfs[i][other_path_type][steps[i]].col;

        // [2-4] If other robot is at the end of its path, assign its current location
        if (other_robot_next_row == -1 && other_robot_next_col == -1) {
                other_robot_next_row = other_robot->row;
                other_robot_next_col = other_robot->col;
        }
}
```

[2-5]현재 이동하고자 하는 로봇의 다음 이동 위치와 다른 로봇의 다음 이동 위치를 계산 후에, 다음 조건식을 검사함.

Condition 1. r->priority > other_robot->priority

: 로봇 구조체의 멤버 변수로 int priority를 도입하여 물류 로봇들이 동시에 같은 위치로 이동하려고 할 때, 높은 priority(낮은 값일수록 높음)를 가진 로봇에게 이동 명령을 전달하고 낮은 priority를 가진 로봇에게 충돌 발생을 일으킴.

Condition 2. next_row == other_robot_next_row
&& next_col == other_robot_next_col

: 현재 이동하고자 하는 로봇의 다음 이동 위치와 다른 로봇의 다음 이동 위치가 동일하다면 충돌 발생을 일으킴.

두 조건식이 True일 경우 1을 반환하여 충돌이 발생함을 알림. [3]그렇지 않은 경우, 0을 반환하여 충돌이 발생하지 않음을 알림.

```
// [2-5] Collision occurs if next location of current robot is same as next location of other robot
if (r->priority > other_robot->priority && next_row == other_robot_next_row && next_col == other_robot_next_col) {
        return 1;
}
```

```
// [3] No collision
return 0;
```

b. 결과

실행 출력 결과는 /output.txt 에 저장함. 각 스텝마다 print_map 함수에 의해 출력된 맵 정보와 각 place별로 위치해 있는 로봇들의 정보를 출력함. 그 다음으로 STEP 값을 출력, 중앙 관제 노드가 로봇들로부터 받은 message에 따라 로봇의 상태를 인자 값으로 하여 print_robot_current_status 함수에 의해 출력된 모든 로봇들의 상태를 출력함.

```
ram@DESKTOP-6FMR5E6:~/os-pintos-project-2025$ cat output.txt
qemu-system-i386 -device isa-debug-exit -hda /tmp/pmQxKJJMyZ.dsk -m 4 -net none -nographic -monitor null
WARNING: Image format was not specified for '/tmp/pmQxKJJMyZ.dsk' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.........
Kernel command line: automated_warehouse 5 2A:4C:2B:2C:3A
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  523,468,800 loops/s.
Boot complete.
arguments list:automated_warehouse, 5, 2A:4C:2B:2C:3A


==================================
*** Automated Warehouse Simulator ***
***        20214234 김하림        ***
==================================


==================================
STEP_INFO_START::0
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                        S    X
X    X    X    X    X    W    X
```

다음은 ./run_warehouse.sh 5 2A:4C:2B:2C:3A 명령을 실행하였을 경우 실행 결과임.

```
ram@DESKTOP-6FMR5E6: ~/os-pintos-project-2025$ cat output.txt
qemu-system-i386 -device isa-debug-exit -hda /tmp/pmQxKJJMyZ.dsk -m 4 -net none -nographic -monitor null
WARNING: Image format was not specified for '/tmp/pmQxKJJMyZ.dsk' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.........
Kernel command line: automated_warehouse 5 2A:4C:2B:2C:3A
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  523,468,800 loops/s.
Boot complete.
arguments list:automated_warehouse, 5, 2A:4C:2B:2C:3A


====================================
*** Automated Warehouse Simulator ***
***        20214234 김하람         ***
====================================
```

```
====================================
STEP_INFO_START::0
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                   S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:R1M0,R2M0,R3M0,R4M0,R5M0,
A:
B:
C:
STEP_INFO_DONE::0

Current Robot Status:
<Robot 1> Row: 0, Col: 0, Payload: 0, Required Payload: 0, Stopped: 0
<Robot 2> Row: 0, Col: 0, Payload: 0, Required Payload: 0, Stopped: 0
<Robot 3> Row: 0, Col: 0, Payload: 0, Required Payload: 0, Stopped: 0
<Robot 4> Row: 0, Col: 0, Payload: 0, Required Payload: 0, Stopped: 0
<Robot 5> Row: 0, Col: 0, Payload: 0, Required Payload: 0, Stopped: 0
```

```
====================================
STEP_INFO_START::1
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                        S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:R1M0,R2M0,R3M0,R4M0,R5M0,
A:
B:
C:
STEP_INFO_DONE::1

Current Robot Status:
<Robot 1> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 2> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 3> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 4> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 5> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::2
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                   R1   X
X    X    X    X    X    W    X

PLACE_INFO::
W:R2M0,R3M0,R4M0,R5M0,
A:
B:
C:
STEP_INFO_DONE::2

Current Robot Status:
<Robot 1> Row: 5, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 3> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 4> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 5> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::3
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                   R1   R2   X
X    X    X    X    X    W    X

PLACE_INFO::
W:R3M0,R4M0,R5M0,
A:
B:
C:
STEP_INFO_DONE::3

Current Robot Status:
<Robot 1> Row: 5, Col: 4, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 5, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 3> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 4> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 5> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::4
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6    R2   X
C              R1        R3   X
X    X    X    X    X    W    X

PLACE_INFO::
W:R4M0,R5M0,
A:
B:
C:
STEP_INFO_DONE::4

Current Robot Status:
<Robot 1> Row: 5, Col: 3, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 4, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 3> Row: 5, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 5> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::5
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5    R2   X
X         3    X    6         X
C         R1        R3   R4   X
X    X    X    X    X    W    X

PLACE_INFO::
W:R5M0,
A:
B:
C:
STEP_INFO_DONE::5

Current Robot Status:
<Robot 1> Row: 5, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 3, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 3> Row: 5, Col: 4, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 5, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 6, Col: 5, Payload: 0, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::6
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4    R2   X
B         2    X    5         X
X         3    X    6         X
C    R1        R3   R4   R5   X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::6

Current Robot Status:
<Robot 1> Row: 5, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 2, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 3> Row: 5, Col: 3, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 5, Col: 4, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 5, Col: 5, Payload: 0, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::7
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    R2        X
B         2    X    5         X
X    R1   3    X    6         X
C         R3   R4   R5   S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::7

Current Robot Status:
<Robot 1> Row: 4, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 2, Col: 4, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 3> Row: 5, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 5, Col: 3, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 5, Col: 4, Payload: 0, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::8
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    R2M1      X
B    R1   2    X    5         X
X         3    X    6         X
C    R3   R4   R5        S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::8

Current Robot Status:
<Robot 1> Row: 3, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 2, Col: 4, Payload: 1, Required Payload: 1, Stopped: 1
<Robot 3> Row: 5, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 5, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 5, Col: 3, Payload: 0, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::9
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4    R2M1 X
B         R1   X    5         X
X    R3   3    X    6         X
C    R4   R5             S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::9

Current Robot Status:
<Robot 1> Row: 3, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 2> Row: 2, Col: 5, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 4, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 5, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 5, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::10
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B    R3   R1M1 X    5    R2M1 X
X    R4   R5   X    6         X
C                   S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::10

Current Robot Status:
<Robot 1> Row: 3, Col: 2, Payload: 1, Required Payload: 1, Stopped: 1
<Robot 2> Row: 3, Col: 5, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 4, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 4, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::11
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B    R1M1 R3   X    5         X
X    R4   R5M1 X    6    R2M1 X
C                        S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::11

Current Robot Status:
<Robot 1> Row: 3, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 2> Row: 4, Col: 5, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 4> Row: 4, Col: 1, Payload: 0, Required Payload: 1, Stopped: 1
<Robot 5> Row: 4, Col: 2, Payload: 1, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::12
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X    R1M1 1    X    4         X
B    R4   R3M1 X    5         X
X    R5M1 3    X    6         X
C                   R2M1 X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::12

Current Robot Status:
<Robot 1> Row: 2, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 2> Row: 5, Col: 5, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 2, Payload: 1, Required Payload: 1, Stopped: 1
<Robot 4> Row: 3, Col: 1, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 4, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::13
MAP_INFO::
X    X    X    X    X    X    X
A    R1M1      7              X
X         1    X    4         X
B    R3M1 R4   X    5         X
X    R5M1 3    X    6         X
C                   R2M1 S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:
B:
C:
STEP_INFO_DONE::13

Current Robot Status:
<Robot 1> Row: 1, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 2> Row: 5, Col: 4, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 4> Row: 3, Col: 2, Payload: 0, Required Payload: 1, Stopped: 0
<Robot 5> Row: 4, Col: 1, Payload: 1, Required Payload: 1, Stopped: 1
```

```
====================================
STEP_INFO_START::14
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B    R5M1 R4M1 X    5         X
X         3    X    6         X
C              R2M1      S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:R1M1,
B:R3M1,
C:
STEP_INFO_DONE::14

Current Robot Status:
<Robot 1> Row: 1, Col: 0, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 2> Row: 5, Col: 3, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 0, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 4> Row: 3, Col: 2, Payload: 1, Required Payload: 1, Stopped: 1
<Robot 5> Row: 3, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::15
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X    R5M1 1    X    4         X
B    R4M1 2    X    5         X
X         3    X    6         X
C         R2M1           S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:R1M0,
B:R3M0,
C:
STEP_INFO_DONE::15

Current Robot Status:
<Robot 1> Row: 1, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 2> Row: 5, Col: 2, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 4> Row: 3, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 5> Row: 2, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::16
MAP_INFO::
X    X    X    X    X    X    X
A    R5M1           7              X
X         1    X    4         X
B         2    X    5         X
X    R4M1 3    X    6         X
C    R2M1                S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:R1M0,
B:R3M0,
C:
STEP_INFO_DONE::16

Current Robot Status:
<Robot 1> Row: 1, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 2> Row: 5, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 4> Row: 4, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 5> Row: 1, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::17
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C    R4M1                S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:R1M0,R5M1,
B:R3M0,
C:R2M1,
STEP_INFO_DONE::17

Current Robot Status:
<Robot 1> Row: 1, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 2> Row: 5, Col: 0, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 3> Row: 3, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 4> Row: 5, Col: 1, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 5> Row: 1, Col: 0, Payload: 1, Required Payload: 1, Stopped: 0
```

```
====================================
STEP_INFO_START::18
MAP_INFO::
X    X    X    X    X    X    X
A              7              X
X         1    X    4         X
B         2    X    5         X
X         3    X    6         X
C                   S    X
X    X    X    X    X    W    X

PLACE_INFO::
W:
A:R1M0,R5M0,
B:R3M0,
C:R2M0,R4M1,
STEP_INFO_DONE::18

Current Robot Status:
<Robot 1> Row: 1, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 2> Row: 5, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 3> Row: 3, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
<Robot 4> Row: 5, Col: 0, Payload: 1, Required Payload: 1, Stopped: 0
<Robot 5> Row: 1, Col: 0, Payload: 0, Required Payload: 0, Stopped: 1
```

```
====================================
*** All robots are done for tasks ***
***   Shutting Down Simulator ..  ***
====================================


Timer: 2027 ticks
Thread: 1900 idle ticks, 127 kernel ticks, 0 user ticks
Console: 14880 characters output
Keyboard: 0 keys pressed
Powering off...
```