Compile Commands

gcc -std=c99 -Wall -Werror worms.c -o worms.exe

There is also a Makefile which does the same thing.

Code Logic

```c
int find_node_index(struct DoublyList* list, int x, int y)
{
    if(list == NULL || list->head == NULL) return -1;

    struct Node *current = list->head;
    for (int i = 0; i < list->elemcount; i++)
    {
        if (current->data->x == x && current->data->y == y) return i;

        current = current->next;
    }

    return -1;
}
```

First of all, I created this helper method to get the Node indexes.

```c
printf("Enter x y coordinates to attack\n");
int x, y;
scanf("%d %d", &x, &y);

int hit_index = -1;
int hit_worm_index = -1;

for (int i = 0; i < wormcount; i++)
{
    hit_index = find_node_index(wormfield[i], x, y);
    if (hit_index != -1)
    {
        hit_worm_index = i;
        break;
    }
}
```

In here, I checked if any worm got hit and if it did where did it get hit and which one got hit.

```c
if(hit_worm_index == -1)
{
    printf("Hit missed!\n");
}
```

If it didn’t get hit, I printed this message.

```c
else
{
    printf("Hit! Worm %d at position %d\n", hit_worm_index, hit_index);

    int length = wormfield[hit_worm_index]->elemcount;

    int remove_start = hit_index - 1;
    int remove_end = hit_index + 1;

    if(remove_start < 0) remove_start = 0;
    if(remove_end >= length) remove_end = length - 1;

    int left_remaining = remove_start;
    int right_remaining = length - (remove_end + 1);
    printf("Debug: remove_start=%d, remove_end=%d\n", remove_start, remove_end);
    printf("Debug: left_remaining=%d, right_remaining=%d\n", left_remaining, right_remaining);
```

If it got hit, I printed the worm that got hit and the position. I calculated the the positions to start and end removing from the worm, and the lengths of those.

```c
if (left_remaining == 0 && right_remaining == 0)
{
    // worm öldü
    printf("Worm died completely!\n");

    while (wormfield[hit_worm_index]->head != NULL)
    {
        removeFront(wormfield[hit_worm_index]);
    }
    free(wormfield[hit_worm_index]);
    wormfield[hit_worm_index] = NULL;

    for (int i = hit_worm_index; i < wormcount - 1; i++)
    {
        wormfield[i] = wormfield[i + 1];
    }
    wormfield[wormcount - 1] = NULL;
    wormcount--;
}
```

If neither the left or right part survived, I killed the worm.

```
else if(left_remaining <= 0 && right_remaining > 0)
{
    // worm'ün sol kısmı gitti
    printf("Left part destroyed, right part survives\n");

    for (int i = 0; i <= remove_end; i++)
    {
        removeFront(wormfield[hit_worm_index]);
    }
}
else if(left_remaining > 0 && right_remaining <= 0)
{
    // worm'ün sağ kısmı gitti
    printf("Right part destroyed, left part survives\n");

    for (int i = remove_end; i >= remove_start; i--)
    {
        removeBack(wormfield[hit_worm_index]);
    }
}
```

If either the left or right part got destroyed but the other part survived, I just removed the dead part.

```
else
{
    // worm ikiye ayrıldı
    printf("Worm split into two parts!\n");

    struct DoublyList* new_worm = (struct DoublyList*) malloc(sizeof(struct DoublyList));
    new_worm->head = NULL;
    new_worm->tail = NULL;
    new_worm->elemcount = 0;

    struct Node* position_ptr = wormfield[hit_worm_index]->head;
    for (int i = 0; i <= remove_end; i++)
    {
        position_ptr = position_ptr->next;
    }

    while (position_ptr != NULL)
    {
        struct WormPart* new_part = malloc(sizeof(struct WormPart));
        new_part->x = position_ptr->data->x;
        new_part->y = position_ptr->data->y;

        addBack(new_worm, new_part);
        position_ptr = position_ptr->next;
    }

    if(new_worm->elemcount > 0)
    {
        wormfield[wormcount] = new_worm;
        wormcount++;
    }
    else
    {
        free(new_worm);
    }

    for (int i = 0; i < length - left_remaining; i++)
    {
        removeBack(wormfield[hit_worm_index]);
    }
}
```

If it was none of the above cases, it meant that the worm will still have two parts, therefore it should be split in half. So I created the right part as a new worm, and removed every part starting from the first part that will be removed from the original worm, therefore I split the worm in half.