```c
#include <stdio.h>
#include <stdlib.h>
#define BLOCK_SIZE 256
struct Block {
    char data[BLOCK_SIZE];
    struct Block* next;
};
int main() {
    struct Block* firstBlock = NULL;
    struct Block* lastBlock = NULL;
    int blockCount = 0;
    int blockNumber;
    char data[BLOCK_SIZE];
    char choice;
    printf("Linked Allocation Simulation\n");
    while (1) {
        printf("Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: ");
        scanf(" %c", &choice);
        if (choice == 'Q' || choice == 'q') {
            break;
        } else if (choice == 'W' || choice == 'w') {
            printf("Enter data for the block: ");
            scanf(" %[^\n]", data);
            struct Block* newBlock = (struct Block*)malloc(sizeof(struct Block));
            if (!newBlock) {
                printf("Memory allocation failed!\n");
                return 1;
            }
            for (int i = 0; i < BLOCK_SIZE; i++) {
                newBlock->data[i] = data[i];
            }
            newBlock->next = NULL;
            if (blockCount == 0) {
                firstBlock = newBlock;
                lastBlock = newBlock;
```

```
        } else {
            lastBlock->next = newBlock;
            lastBlock = newBlock;
        }
        blockCount++;
    } else if (choice == 'R' || choice == 'r') {
        if (blockCount == 0) {
            printf("No blocks to read.\n");
            continue;
        }
        printf("Enter the block number to read (1-%d): ", blockCount);
        scanf("%d", &blockNumber);
        if (blockNumber < 1 || blockNumber > blockCount) {
            printf("Invalid block number. The valid range is 1-%d.\n", blockCount);
        } else {
            struct Block* currentBlock = firstBlock;
            for (int i = 1; i < blockNumber; i++) {
                currentBlock = currentBlock->next;
            }
            printf("Block %d Data: %s\n", blockNumber, currentBlock->data);
        }
    } else {
        printf("Invalid choice. Please enter W, R, or Q.\n");
    }
}
struct Block* currentBlock = firstBlock;
while (currentBlock != NULL) {
    struct Block* nextBlock = currentBlock->next;
    free(currentBlock);
    currentBlock = nextBlock;
}
return 0;
}
```

```
Linked Allocation Simulation
Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: w
Enter data for the block: hii this is a sample
Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: w
Enter data for the block: yeahhhhh
Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: r
Enter the block number to read (1-2): 2
Block 2 Data: yeahhhhh
Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: r
Enter the block number to read (1-2): 1
Block 1 Data: hii this is a sample
Enter 'W' to write a block, 'R' to read a block, or 'Q' to quit: q

--------------------------------
Process exited after 256.1 seconds with return value 0
Press any key to continue . . .
```