```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Block{
    char name; // Name of the memory block
    int start; // Start of the memory block
    int end; // End of the memory block
    int valid; // If this block is valid or has it been freed or leaked?
}Block;

int memSize = 1000;
int *mem;
Block *blocks;
int numBlocks = 0;
int memInUse = 0;

// Check if this identifier has currently been allocated memory or not
int checkExistingAllocation(char iden){
    for(int i = 0; i < numBlocks; i++){
        if(blocks[i].valid == 1 && blocks[i].name == iden)
            return i;
    }
    return -1;
}

int doFree(char iden){
    int idxBlock = checkExistingAllocation(iden);
    if(idxBlock == -1)
        return 0; // Failure - nothing to free here

    int start = blocks[idxBlock].start, end = blocks[idxBlock].end;
    int size = end - start + 1;
    memInUse -= size; // Memory in use goes down

    for(int i = start; i <= end; i++)
        mem[i] = 0; // Free these bytes

    blocks[idxBlock].valid = 0; // No longer a valid block
    return 1; // Successful free
}

// Try to get a new contiguous block of free memory
// Return its starting index if successful
int getNewBlock(int size, int start){
    if(start + size - 1 >= memSize)
        return -1; // Reached the end of memory - not enough space
    for(int i = start; i < start + size; i++){
        if(mem[i] == 1){ // Too bad - try the next block
            while(mem[i] == 1) // Find the next empty byte
                i++;
            return getNewBlock(size, i);
        }
    }
    // We were successful in finding a block from start itself
    for(int i = start; i < start + size; i++)
        mem[i] = 1;
    return start; // Return the starting position of the new block
}

int doMalloc(char iden, int size){
    // Allocate a new contiguous block of this size and get its index
    int memIdx = getNewBlock(size, 0);
    if(memIdx < 0)
        return 0;

    memInUse += size; // Memory in use goes up (even if there is leak)

    // Has this identifier already been allocated memory in an old block?
    int blockIdx = checkExistingAllocation(iden);
```

```
    if(blockIdx >= 0){
        printf("MEMORY LEAK\n");
        blocks[blockIdx].valid = 0; // Not a valid block anymore
        // However, this memory is not freed - it gets leaked :(
    }

    // Create a new block for the newly allocated memory
    Block newBlock;
    newBlock.name = iden;
    newBlock.start = memIdx;
    newBlock.end = memIdx + size - 1;
    newBlock.valid = 1;

    // Put this new block into the list of Blocks
    blocks[numBlocks++] = newBlock;
    return 1;    // Successful memory allocation
}

int main(){
    int N;
    scanf("%d\n", &N);
    mem = (int*)calloc(memSize, sizeof(int)); // Initialize memory to 0
    blocks = (Block*)malloc(N * sizeof(Block));

    char command[10], iden;
    int size, isSuccess;
    for(int c = 0; c < N; c++){
        scanf("%s %c", command, &iden);
        switch(command[0]){
            case 'm':
                scanf("%d", &size);
                isSuccess = doMalloc(iden, size);
                break;
            case 'f':
                isSuccess = doFree(iden);
                break;
        }
        if(isSuccess)
            printf("SUCCESS\n");
        else
            printf("FAILURE\n");
    }
    printf("%d", memInUse);
    return 0;
}
```