Elevator System

Our project is the implementation for internal controls of an elevator.
We have implemented it using an **ordered doubly linked list**.
Given below are the descriptions of our structures and functions:

STRUCTURES:

struct button	The node structure for our doubly linked list
int press	The desired destination floor number (floors are from 0-9)
int flooroc	Number of people getting off on that floor
struct button* next, *prev	Links to the next and previous nodes
struct elevator	Holds the details of our elevator
int occupants	Total number of people in the elevator
int* floordisp	An array to keep track of which floor buttons are pressed. Its use is similar to that of the "visited" array used with graphs
button* head,*tail	Head and Tail of our list
char dxn	Determines which way to traverse through list: - forward traversal if dxn is 'u' (up)

FUNCTIONS:

1. button* createNode(int press,int flooroc)
   creates and returns a node with values as passed in arguments.
   if the button pressed is a number button, it will be converted to integer before inserting into the list.

2. char pressButton()
   accepts and returns elevator button to be pressed from the user.

3. void delNode(elevator* e,button* b)
   deletes from the list the node that is passed as argument (button* b).
   e->floordisp is updated.

4. int insert(elevator* e,button* b)
   performs ordered insertion into the list the node that is passed as argument (button* b).
   (insertion occurs such that the list is sorted in ascending order of floor number).
   e->floordisp is updated.

5. void printgrid(int curfloor,elevator* e)
   displays the button grid of the elevator.
   displays current floor in blue.
   displays floors to be visited (those buttons have been pressed) in yellow.
   uses e->floordisp array to determine which numbers to display in color.
   also displays number of elevator occupants at the bottom, as little cheering people -> \O/

6. void delayrun(int milliseconds)
   To more realistically simulate the working of an elevator, we have the execution pause for a certain number of seconds at certain points, like when the elevator doors are closing, or the elevator is moving.
   We achieve this using the delay function.

7. void countdelay(int curfloor,int nextfloor)
   The more floors the elevator has to travel through to reach its destination floor, the longer it will take to do so.
   We use this function to determine how long to stall execution, as the elevator moves from one floor to another.
   We stall by 1 second per floor the elevator has to move.

8. void enterLift(elevator* e)
   When the elevator stops on a certain floor, this function accepts a set of (floor) button presses, along with the number of people (flooroc) getting down on that corresponding floor.
   It inserts these (floor,flooroc) pairs into the linked list.
   Function keeps accepting the (floor,flooroc) pairs until maximum capacity of the elevator is reached, or the "close door" button (x) is pressed.
   Either of these conditions will prompt the elevator to stop moving.
   If the "alert" button (!) is pressed, it will trigger an alert and shut down the elevator, terminating the program.

9. void move(elevator* e)
   Handles moving of the elevator from one floor to another, taking care to handle the direction of movement, and change in direction when required.
   Depending on whether it is moving up or down, the function traverses the doubly linked list
   forwards (up), or backwards (down). Uses switch statement to determine this.
   Once the elevator reaches a certain floor, the function deletes that node from the list, and moves on to the node of next floor to be visited.

10. int main()
    a short, clean function which sets the head and tail of the doubly linked list with a (0,0)

(floor, flooroc) node.
Sets the direction of the elevator to 'u' (up), initially.
Creates e->floordisp array of size 10 (floors 0-9), and initializes all values to 0 using calloc.
Calls the move function, starts up our virtual elevator.