

// C Code For Mentioned Problem //

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

//structure named as node is created for car of the train

```
struct node
{
    int passengers;
    //character array of size 100 is declared for storing name of the car
    char name[100];
    struct node *next;
};
```

// createnode() function is defined which creates a new node and initializes it with given values to the function

```
struct node* createnode(int d,char name[])
{
    //new node is created using malloc() function and address is stored in newnode pointer
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    //given values are initialized to the function
    newnode->passengers=d;
    //strcpy() fuction is used for assigning string
    strcpy(newnode->name,name);
    newnode->next=NULL;

    //returned address of newly created node
    return newnode;
}
```

// display()

//display() method is defined which prints the information of each car in the list

```
void display(struct node *head)
{
    //if list is empty
    if(head==NULL)
    {
        printf("\nList is Empty\n");
        return;
    }

    //created temp pointer and stored address of head node
    struct node *temp=head;

    //traversed till temp is not equal to null
    while(temp!=NULL)
    {
        //printed passengers and name from temp node
        printf("\n%d %s",temp->passengers,temp->name);
        //updated temp incrementing it by 1 position
        temp=temp->next;
    }
}

//display method ended
```

// length()

//length() function is defined which returns the number of nodes present in the list

```
int length(struct node *head)
{
    //declared a counter c and initialized it with 0
    int c=0;
    //declared temp pointer and assigned head address to it
    struct node *temp=head;
    //traversed till temp is not equal to null
    while(temp!=NULL)
    {
        //increment c by 1
        c++;
        //update temp incrementing by one position
        temp=temp->next;
    }
    //returned counter c
    return c;
}

//length() method ended
```

//addCar()

// addCar() method is defined which adds a new node in the ens of the list

```
void addCar(struct node **head,int d,char name[])
{
    //newnode and temp pointers are declared
    struct node *newnode,*temp;

    //new node is created using createnode() function and returned address is stored in newnode pointer
    newnode =createnode(d,name);

    //if list is empty then add newnode as first node of the list
    if(*head==NULL)
    {
        *head=newnode;
        return;
    }

    //*/head is assigned to temp pointer
```

```

temp=*head;
//traversed while next of temp is not equal to null
//( traversed till last node )
while(temp->next!=NULL)
    temp=temp->next;

//added new node at the end of list
temp->next=newnode;
}
//addCar() function ended

```

// removeCars()

```

//removeCars() function is defined below
//it removes nodes where number of passengers are 0
void removeCars(struct node **head)
{
    //two pointers named as prev and temp are declared and initialized with *head
    //prev will point one node before the temp in the list
    struct node *prev,*temp;
    temp = *head;
    prev = *head;

    //traversed till temp is not equal to NULL
    while(temp!=NULL)
    {
        //if passengers in temp node is equal to 0
        if(temp->passengers == 0 )
        {
            //checked if temp is first node or not
            if(temp == *head)
            {
                printf("\n%d %s removed ",temp->passengers,temp->name);
                //if it is first node then make second node as head node
                *head = temp->next;
                //free memory for temp node using free() function
                free(temp);
                //re initialized temp and prev pointer
                temp = *head;
                prev = *head;
            }

            //if temp is not the first node then
            else
            {
                //remove temp node by changing the address field of the prev node
                printf("\n%d %s removed ",temp->passengers,temp->name);
                prev->next = temp->next;
                free(temp);
                temp = prev->next;
            }
        }

        //if passengers in temp node is not equal to 0
        else
        {
            //increment prev
            prev = temp;
            //increment temp
            temp = temp->next;
        }
    }
}
//removeCars() function ended

```

//main()

```

//main() function is defined below
int main()
{
    //head node is created and initialized to NULL
    struct node *head=NULL;

    //cars are added in the list using addCar() fucntion
    addCar(&head,83,"B1");
    addCar(&head,72,"B2");
    addCar(&head,0,"B3");
    addCar(&head,69,"B4");
    addCar(&head,0,"B5");

    printf("Cars in the list:");
    //display() fucntion is called
    display(head);

    //removeCars() fucntion is called
    printf("\nremoveCars() fucntion is called ");
    removeCars(&head);
    //after removing empty cars ,display() fucntion is called
    printf("\nAfter removing empty cars:");
    display(head);
    //length() function is called and returned length of list is stored in l variable
    int l = length(head);
    //length of the list is printed
    printf("\nNumber of cars left in Train : %d\n",l);

    return 0;
}
//main() method ended

```

main.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  //structure named as node is created for car of the train
5  struct node
6  {
7      int passengers;
8      //character array of size 100 is declared for storing name of the car
9      char name[100];
10     struct node *next;
11 };
12
13 // createnode() function is defined which creates a new node and initializes it with given values to the function
14 struct node* createnode(int d,char name[])
15 {
16     //new node is created using malloc() function and address is stored in newnode pointer
17     struct node *newnode=(struct node *)malloc(sizeof(struct node));
18     //given values are initialized to the function
19     newnode->passengers=d;
20     //strcpy() function is used for assigning string
21     strcpy(newnode->name,name);
22     newnode->next=NULL;
23
24     //returned address of newly created node
25     return newnode;
26 }
27
28 //display() method is defined which prints the information of each car in the list
29 void display(struct node *head)
30 {
31     //if list is empty
32     if(head==NULL)
33     {
34         printf("\nList is Empty\n");
35         return;
36     }
37
38     //created temp pointer and stored address of head node
39     struct node *temp=head;
40
41     //traversed till temp is not equal to null
42     while(temp!=NULL)
43     {
44         //printed passengers and name from temp node
45         printf("\n%d %s",temp->passengers,temp->name);
46         //updated temp incrementing it by 1 position
47         temp=temp->next;
48     }
49 }
50
51 //display method ended
52
```

```

53
54 //length() function is defined which returns the number of nodes present in the List
55 int length(struct node *head)
56 {
57     //declared a counter c and initialized it with 0
58     int c=0;
59     //declared temp pointer and assigned head address to it
60     struct node *temp=head;
61     //traversed till temp is not equal to null
62     while(temp!=NULL)
63     {
64         //increment c by 1
65         c++;
66         //update temp incrementing by one position
67         temp=temp->next;
68     }
69     //returned counter c
70     return c;
71 }
72 //length() method ended
73
74
75 // addCar() method is defined which adds a new node in the end of the List
76 void addCar(struct node **head,int d,char name[])
77 {
78     //newnode and temp pointers are declared

```

```

79     struct node *newnode,*temp;
80
81     //new node is created using createnode() function and returned address is stored in newnode pointer
82     newnode =createnode(d,name);
83
84     //if List is empty then add newnode as first node of the List
85     if(*head==NULL)
86     {
87         *head=newnode;
88         return;
89     }
90
91     // *head is assigned to temp pointer
92     temp=*head;
93     //traversed while next of temp is not equal to null
94     //( traversed till last node )
95     while(temp->next!=NULL)
96         temp=temp->next;
97
98     //added new node at the end of List
99     temp->next=newnode;
100 }
101 //addCar() function ended
102
103
104

```

```

105 //removeCars() function is defined below
106 //it removes nodes where number of passengers are 0
107 void removeCars(struct node **head)
108 {
109     //two pointers named as prev and temp are declared and initialized with *head
110     //prev will point one node before the temp in the list
111     struct node *prev,*temp;
112     temp = *head;
113     prev = *head;
114
115     //traversed till temp is not equal to NULL
116     while(temp!=NULL)
117     {
118         //if passengers in temp node is equal to 0
119         if(temp->passengers == 0 )
120         {
121             //checked if temp is first node or not
122             if(temp == *head)
123             {
124                 printf("\n%d %s removed ",temp->passengers,temp->name);
125                 //if it is first node then make second node as head node
126                 *head = temp->next;
127                 //free memory for temp node using free() function
128                 free(temp);
129                 //re initialized temp and prev pointer
130                 temp = *head;

```

```

131                 prev = *head;
132
133             }
134
135             //if temp is not the first node then
136             else
137             {
138                 //remove temp node by changing the address field of the prev node
139                 printf("\n%d %s removed ",temp->passengers,temp->name);
140                 prev->next = temp->next;
141                 free(temp);
142                 temp = prev->next;
143
144             }
145
146         }
147
148         //if passengers in temp node is not equal to 0
149         else
150         {
151             //increment prev
152             prev = temp;
153             //increment temp
154             temp = temp->next;
155         }
156

```



```

157     }
158
159 }
160 //removeCars() function ended
161
162
163
164 //main() function is defined below
165 int main()
166 {
167     //head node is created and initialized to NULL
168     struct node *head=NULL;
169
170     //cars are added in the list using addCar() fucntion
171     addCar(&head,83,"B1");
172     addCar(&head,72,"B2");
173     addCar(&head,0,"B3");
174     addCar(&head,69,"B4");
175     addCar(&head,0,"B5");
176
177     printf("Cars in the list:");
178     //display() fucntion is called
179     display(head);
180
181     //removeCars() fucntion is called
182     printf("\nremoveCars() fucntion is called ");

```

```

182     printf("\nremoveCars() fucntion is called ");
183     removeCars(&head);
184     //after removing empty cars ,display() fucntion is called
185     printf("\nAfter removing empty cars:");
186     display(head);
187     //length() function is called and returned length of list is stored in L variable
188     int l = length(head);
189     //length of the list is printed
190     printf("\nNumber of cars left in Train : %d\n",l);
191
192     return 0;
193
194 }
195 //main() fucntion ended

```

Output Windnow 1:

☐
☐
☐

input

```

Cars in the list:
83 B1
72 B2
0 B3
69 B4
0 B5
removeCars() fucntion is called
0 B3 removed
0 B5 removed
After removing empty cars:
83 B1
72 B2
69 B4
Number of cars left in Train : 3

...Program finished with exit code 0
Press ENTER to exit console.

```

Output Window 2:

```
input
Cars in the list:
0 B1
0 B2
0 B3
0 B4
0 B5
removeCars() fucntion is called
0 B1 removed
0 B2 removed
0 B3 removed
0 B4 removed
0 B5 removed
After removing empty cars:
List is Empty

Number of cars left in Train : 0

...Program finished with exit code 0
Press ENTER to exit console.
```

Please Upvote If You Liked This Answer :)