

Soluzione Esercizio 1 versione Lista di Liste

```
struct Magazzino {
    int id;
    struct Camion* elenco;
    struct Magazzino* next;
};

struct Camion {
    int dataOra; /*gestito come timestamp*/
    int codiceAzienda;
    int tipoMerce; /*1 verdura, 2 latte, 3 frutta*/
    float peso;
    struct Camion* next;
};

struct Magazzino* elencoMagazzini = NULL;

/*
VERSIONE 1
si suppone che il magazzino con il codice passato esista in elenco
(versione che utilizza il comando break)
*/
void registraCamion(struct Magazzino* elencoMagazzini, int idMagazzino,
                   int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    struct Magazzino* tmp;
    struct Camion* nuovo, prev, aux;

    tmp = elencoMagazzini;
    while (tmp != NULL) {
        if (tmp->id == idMagazzino) {
            nuovo = (struct Camion*)malloc(sizeof(struct Camion));
            nuovo->next = NULL;
            nuovo->dataOra = dataOra;
            nuovo->codiceAzienda = codiceAzienda;
            nuovo->tipoMerce = tipoMerce;
            nuovo->peso = peso;
            aux = tmp->elenco;
            prev = NULL;
            while (aux != NULL) {
                if (aux->tipoMerce > nuovo->tipoMerce) {
                    nuovo->next = aux;
                    if (prev == NULL) {
                        tmp->elenco = nuovo;
                    }
                    else {
                        prev->next = nuovo;
                    }
                    break;
                }
                prev = aux;
                aux = aux->next;
            }
            if (aux == NULL) {
                if (prev == NULL) {
                    tmp->elenco = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
                break;
            }
        }
    }
}
```

```

        }
    }
    tmp = tmp->next;
}

/*
VERSIONE 2
si suppone che il magazzino con il codice passato esista in elenco
(versione senza il comando break)
*/
void registraCamion(struct Magazzino* elencoMagazzini, int idMagazzino,
                   int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    struct Magazzino* tmp;
    struct Camion* nuovo, prev, aux;
    int inserito = 0;

    tmp = elencoMagazzini;
    while (tmp != NULL) {
        if (tmp->id == idMagazzino) {
            nuovo = (struct Camion*)malloc(sizeof(struct Camion));
            nuovo->next = NULL;
            nuovo->dataOra = dataOra;
            nuovo->codiceAzienda = codiceAzienda;
            nuovo->tipoMerce = tipoMerce;
            nuovo->peso = peso;
            aux = tmp->elenco;
            prev = NULL;
            while (aux != NULL && !inserito) {
                if (aux->tipoMerce > nuovo->tipoMerce) {
                    nuovo->next = aux;
                    if (prev == NULL) {
                        tmp->elenco = nuovo;
                    }
                    else {
                        prev->next = nuovo;
                    }
                    inserito = 1;
                }
                else {
                    prev = aux;
                    aux = aux->next;
                }
            }
            if (!inserito) {
                if (prev == NULL) {
                    tmp->elenco = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
            }
            tmp = NULL;
        }
        else {
            tmp = tmp->next;
        }
    }
}

```

```

/*
VERSIONE 3
versione che aggiunge il magazzino in caso non sia presente in elenco
*/
void registraCamion(struct Magazzino** elencoMagazzini, int idMagazzino,
                    int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    struct Magazzino* tmp, magRegistrazione = NULL;
    struct Camion* nuovo, prev, aux;

    tmp = *elencoMagazzini;
    while (tmp != NULL && magRegistrazione == NULL) {
        if (tmp->id == idMagazzino) {
            magRegistrazione = tmp;
        }
        else {
            tmp = tmp->next;
        }
    }

    if (magRegistrazione == NULL) {
        magRegistrazione = (struct Magazzino*)malloc(sizeof(struct
Magazzino));
        magRegistrazione->elenco = NULL;
        magRegistrazione->id = idMagazzino;
        magRegistrazione->next = *elencoMagazzini;
        *elencoMagazzini = magRegistrazione;
    }

    nuovo = (struct Camion*)malloc(sizeof(struct Camion));
    nuovo->next = NULL;
    nuovo->dataOra = dataOra;
    nuovo->codiceAzienda = codiceAzienda;
    nuovo->tipoMerce = tipoMerce;
    nuovo->peso = peso;
    aux = magRegistrazione->elenco;
    prev = NULL;
    while (aux != NULL && !inserito) {
        if (aux->tipoMerce > nuovo->tipoMerce) {
            nuovo->next = aux;
            if (prev == NULL) {
                magRegistrazione->elenco = nuovo;
            }
            else {
                prev->next = nuovo;
            }
            inserito = 1;
        }
        else {
            prev = aux;
            aux = aux->next;
        }
    }
    if (!inserito) {
        if (prev == NULL) {
            magRegistrazione->elenco = nuovo;
        }
        else {
            prev->next = nuovo;
        }
    }
}

```

```
}
```

```
int contaCamion(struct Magazzino* elencoMagazzini, int tipoMerce) {  
    struct Magazzino* tmp;  
    struct Camion* aux;  
    int conta = 0;  
  
    tmp = elencoMagazzini;  
    while (tmp != NULL) {  
        aux = tmp->elenco;  
        while (aux != NULL) {  
            if (aux->tipoMerce == tipoMerce) {  
                conta++;  
            }  
            aux = aux->next;  
        }  
        tmp = tmp->next;  
    }  
  
    return conta;  
}
```

Soluzione Esercizio 1 versione Array di Liste

```
struct Magazzino {
    int id;
    struct Camion* elenco;
};

struct Camion {
    int dataOra; /*gestito come timestamp*/
    int codiceAzienda;
    int tipoMerce; /*1 verdura, 2 latte, 3 frutta*/
    float peso;
    struct Camion* next;
};

struct Magazzino elencoMagazzini[N];

/*
VERSIONE 1
si suppone che il magazzino con il codice passato esista in elenco
(versione che utilizza il comando break)
*/
void registraCamion(struct Magazzino elencoMagazzini[N], int idMagazzino,
                   int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    int i;
    struct Camion* nuovo, prev, aux;

    for (i = 0; i < N; i++) {
        if (elencoMagazzini[i].id == idMagazzino) {
            nuovo = (struct Camion*)malloc(sizeof(struct Camion));
            nuovo->next = NULL;
            nuovo->dataOra = dataOra;
            nuovo->codiceAzienda = codiceAzienda;
            nuovo->tipoMerce = tipoMerce;
            nuovo->peso = peso;
            aux = elencoMagazzini[i].elenco;
            prev = NULL;
            while (aux != NULL) {
                if (aux->tipoMerce > nuovo->tipoMerce) {
                    nuovo->next = aux;
                    if (prev == NULL) {
                        elencoMagazzini[i].elenco = nuovo;
                    }
                    else {
                        prev->next = nuovo;
                    }
                    break;
                }
                prev = aux;
                aux = aux->next;
            }
            if (aux == NULL) {
                if (prev == NULL) {
                    elencoMagazzini[i].elenco = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
            }
        }
    }
}
```

```

        }
        break;
    }
}

/*
VERSIONE 2
si suppone che il magazzino con il codice passato esista in elenco
(versione che NON utilizza il comando break)
*/
void registraCamion(struct Magazzino elencoMagazzini[N], int idMagazzino,
    int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    int i, inserito = 0;
    struct Camion* nuovo, prev, aux;

    for (i = 0; i < N; i++) {
        if (elencoMagazzini[i].id == idMagazzino) {
            nuovo = (struct Camion*)malloc(sizeof(struct Camion));
            nuovo->next = NULL;
            nuovo->dataOra = dataOra;
            nuovo->codiceAzienda = codiceAzienda;
            nuovo->tipoMerce = tipoMerce;
            nuovo->peso = peso;
            aux = elencoMagazzini[i].elenco;
            prev = NULL;
            inserito = 0;
            while (aux != NULL && !inserito) {
                if (aux->tipoMerce > nuovo->tipoMerce) {
                    nuovo->next = aux->next;
                    if (prev == NULL) {
                        elencoMagazzini[i].elenco = nuovo;
                    }
                    else {
                        prev->next = nuovo;
                    }
                    inserito = 1;
                }
                else {
                    prev = aux;
                    aux = aux->next;
                }
            }
            if (!inserito) {
                if (prev == NULL) {
                    elencoMagazzini[i].elenco = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
            }
            i = N;
        }
    }
}

```

```

/*
VERSIONE 3
la funzione restituisce 0 nel caso in cui il magazzino con il codice fornito non
sia presente, 1 altrimenti
(versione che NON utilizza il comando break)
*/
int registraCamion(struct Magazzino elencoMagazzini[N], int idMagazzino,
                  int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    int i, inserito = 0;
    struct Camion* nuovo, prev, aux;

    for (i = 0; i < N; i++) {
        if (elencoMagazzini[i].id == idMagazzino) {
            nuovo = (struct Camion*)malloc(sizeof(struct Camion));
            nuovo->next = NULL;
            nuovo->dataOra = dataOra;
            nuovo->codiceAzienda = codiceAzienda;
            nuovo->tipoMerce = tipoMerce;
            nuovo->peso = peso;
            aux = elencoMagazzini[i].elenco;
            prev = NULL;
            inserito = 0;
            while (aux != NULL && !inserito) {
                if (aux->tipoMerce > nuovo->tipoMerce) {
                    nuovo->next = aux;
                    if (prev == NULL) {
                        elencoMagazzini[i].elenco = nuovo;
                    }
                    else {
                        prev->next = nuovo;
                    }
                    inserito = 1;
                }
                else {
                    prev = aux;
                    aux = aux->next;
                }
            }
            if (!inserito) {
                if (prev == NULL) {
                    elencoMagazzini[i].elenco = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
            }
            i = N;
        }
    }

    return inserito;
}

```

```
int contaCamion(struct Magazzino elencoMagazzini[N], int tipoMerce) {
    struct Camion* aux;
    int conta = 0, i;

    for (i = 0; i < N; i++) {
        aux = elencoMagazzini[i].elenco;
        while (aux != NULL) {
            if (aux->tipoMerce == tipoMerce) {
                conta++;
            }
            aux = aux->next;
        }
    }

    return conta;
}
```

Soluzione Esercizio 1 versione Array di Liste dove l'id del magazzino corrisponde all'indice della cella all'interno dell'array

```
struct Camion {
    int dataOra; /*gestito come timestamp*/
    int codiceAzienda;
    int tipoMerce; /*1 verdura, 2 latte, 3 frutta*/
    float peso;
    struct Camion* next;
};

struct Camion* elencoMagazzini[N];

/*
VERSIONE 1
si suppone che il magazzino con il codice passato esista in elenco
(versione che utilizza il comando break)
*/
void registraCamion(struct Camion* elencoMagazzini[N], int idMagazzino,
                   int dataOra, int codiceAzienda, int tipoMerce, float peso) {

    struct Camion* nuovo, prev, aux;

    if (idMagazzino < N) {
        nuovo = (struct Camion*)malloc(sizeof(struct Camion));
        nuovo->next = NULL;
        nuovo->dataOra = dataOra;
        nuovo->codiceAzienda = codiceAzienda;
        nuovo->tipoMerce = tipoMerce;
        nuovo->peso = peso;
        aux = elencoMagazzini[idMagazzino];
        prev = NULL;
        while (aux != NULL) {
            if (aux->tipoMerce > nuovo->tipoMerce) {
                nuovo->next = aux->next;
                if (prev == NULL) {
                    elencoMagazzini[idMagazzino] = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
                break;
            }
            prev = aux;
            aux = aux->next;
        }
        if (aux == NULL) {
            if (prev == NULL) {
                elencoMagazzini[idMagazzino] = nuovo;
            }
            else {
                prev->next = nuovo;
            }
        }
    }
}
```

```

/*
VERSIONE 2
si suppone che il magazzino con il codice passato esista in elenco
(versione che NON utilizza il comando break)
*/
void registraCamion(struct Camion* elencoMagazzini[N], int idMagazzino,
                   int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    int inserito = 0;
    struct Camion* nuovo, prev, aux;

    if (idMagazzino < N) {
        nuovo = (struct Camion*)malloc(sizeof(struct Camion));
        nuovo->next = NULL;
        nuovo->dataOra = dataOra;
        nuovo->codiceAzienda = codiceAzienda;
        nuovo->tipoMerce = tipoMerce;
        nuovo->peso = peso;
        aux = elencoMagazzini[idMagazzino];
        prev = NULL;
        inserito = 0;
        while (aux != NULL && !inserito) {
            if (aux->tipoMerce > nuovo->tipoMerce) {
                nuovo->next = aux;
                if (prev == NULL) {
                    elencoMagazzini[idMagazzino] = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
                inserito = 1;
            }
            else {
                prev = aux;
                aux = aux->next;
            }
        }
        if (!inserito) {
            if (prev == NULL) {
                elencoMagazzini[idMagazzino] = nuovo;
            }
            else {
                prev->next = nuovo;
            }
        }
    }
}

```

```

/*
VERSIONE 3
la funzione restituisce 0 nel caso in cui il magazzino con il codice fornito non
sia presente, 1 altrimenti
(versione che NON utilizza il comando break)
*/
int registraCamion(struct Camion* elencoMagazzini[N], int idMagazzino,
                  int dataOra, int codiceAzienda, int tipoMerce, float peso) {
    int inserito = 0;
    struct Camion* nuovo, prev, aux;

    if (idMagazzino < N) {
        nuovo = (struct Camion*)malloc(sizeof(struct Camion));
        nuovo->next = NULL;
        nuovo->dataOra = dataOra;
        nuovo->codiceAzienda = codiceAzienda;
        nuovo->tipoMerce = tipoMerce;
        nuovo->peso = peso;
        aux = elencoMagazzini[idMagazzino];
        prev = NULL;
        inserito = 0;
        while (aux != NULL && !inserito) {
            if (aux->tipoMerce > nuovo->tipoMerce) {
                nuovo->next = aux;
                if (prev == NULL) {
                    elencoMagazzini[idMagazzino] = nuovo;
                }
                else {
                    prev->next = nuovo;
                }
                inserito = 1;
            }
            else {
                prev = aux;
                aux = aux->next;
            }
        }
        if (!inserito) {
            if (prev == NULL) {
                elencoMagazzini[idMagazzino] = nuovo;
            }
            else {
                prev->next = nuovo;
            }
        }
    }
}

```

```
int contaCamion(struct Camion* elencoMagazzini[N], int tipoMerce) {
    struct Camion* aux;
    int conta = 0, i;

    for (i = 0; i < N; i++) {
        aux = elencoMagazzini[i];
        while (aux != NULL) {
            if (aux->tipoMerce == tipoMerce) {
                conta++;
            }
            aux = aux->next;
        }
    }

    return conta;
}
```