LISTE DUBLE

```
struct produs
{
    int *cod;
    char* denumire;
    float pret;
    float cantitate;

};


struct nodls
{
    produs *inf;
    nodls* next, *prev;
};

nodls* inserare(nodls* cap,nodls **coada, produs *p)
{
    nodls* nou = (nodls*)malloc(sizeof(nodls));

    nou->inf = (produs*)malloc(sizeof(produs));
    nou->inf->cod = (int*)malloc(sizeof(int));
    nou->inf->denumire = (char*)malloc((strlen(p-
>denumire)+1)*sizeof(p->denumire));
    *(nou->inf->cod )= *(p->cod);
    strcpy(nou->inf->denumire, p->denumire);
    nou->inf->pret = p->pret;
    nou->inf->cantitate = p->cantitate;
    nou->next = NULL;
    nou->prev = NULL;
    if (cap == NULL)
    {
        cap = nou;
        *coada = nou;
    }
    else{
        nodls* temp = cap;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = nou;
        nou->prev = temp;
        *coada = nou;
    }
    return cap;

}
void traversare(nodls*cap)
{
    nodls* temp = cap;
    while (temp != NULL)
    {
        printf("\n Cod= %d, denumire=%s, pret=%5.2f,
cantittate=%5.2f \n", *(temp->inf->cod), temp->inf-
>denumire, temp->inf->pret, temp->inf->cantitate);
        temp = temp->next;
    }
}

void traversareInversa(nodls* coada)
{
    nodls* temp = coada;
    while (temp != NULL)
    {
        printf("\n Cod= %d, denumire=%s, pret=%5.2f, cantittate=%5.2f \n", *(temp->inf->cod), temp->inf->denumire, temp->inf->pret, temp->inf->cantitate);
        temp = temp->prev;
    }
}
void dezalocare(nodls* cap)
{
    nodls* temp = cap;
    while (temp != NULL)
    {
        nodls* temp2 = temp->next;
        free(temp->inf->denumire);
        free(temp->inf->cod);
        free(temp->inf);
        free(temp);
        temp = temp2;
    }
}
void stergereNodDenumire(nodls **cap,nodls**coada, char *den)
{
    if (strcmp((*cap)->inf->denumire, den) == 0)
    {
        nodls* temp = *cap;
        *cap = (*cap)->next;
        (*cap)->prev = NULL;
        free(temp->inf->denumire);
        free(temp->inf->cod);
        free(temp->inf);
        free(temp);
        return;
    }
    else
        if (strcmp((*coada)->inf->denumire, den) ==
0)
        {
            nodls* temp = *coada;
            *coada = (*coada)->prev;
            (*cap)->next = NULL;
            free(temp->inf->denumire);
            free(temp->inf->cod);
            free(temp->inf);
            free(temp);
            return;
        }
        else
        {
            nodls* temp = *cap;
            while (temp != NULL)
            {
                if (strcmp((temp->inf->denumire),
den) == 0)
                {
                    nodls* urmator = temp->next;
                    nodls* anterior = temp->prev;
                    anterior->next = urmator;
                    urmator->prev = anterior;
                    free(temp->inf->denumire);
                    free(temp->inf->cod);
                    free(temp->inf);
                    free(temp);
                    return;
                }
                else
                {
                    temp = temp->next;
```

```
                        }
                    }

                }

        }



void conversieListaVector(nodls *cap,produs **vect, int *nr)
{
        nodls*temp =cap;
        while (temp != NULL)
        {
                vect[* nr ]= temp->inf;
                (*nr)++;
                nodls *temp2 = temp->next;
                free(temp);
                temp = temp2;

        }


}




void main()
{
        FILE* f = fopen("Fisier.txt", "r");
        int n;
        fscanf(f,"%d", &n);
        nodls* cap = NULL, * coada = NULL;
        produs *p;
        char buffer[20];
        for (int i = 0; i < n; i++)
        {
                p = (produs*)malloc(sizeof(produs));
                p->cod = (int*)malloc(sizeof(int));
                fscanf(f,"%d", p->cod);
                fscanf(f,"%s", buffer);
                p->denumire = (char*)malloc((strlen(buffer) +
1)*sizeof(buffer));
                strcpy(p->denumire, buffer);

                fscanf(f,"%f", &p->pret);

                fscanf(f,"%f", &p->cantitate);
                cap = inserare(cap,&coada, p);
                free(p->denumire);
                free(p->cod);
                free(p);

        }
        fclose(f);

        traversare(cap);
        traversareInversa(coada);
        //dezalocare(cap);
        //dezalocare(coada);
        printf("\n------------------");
        produs** vect =
(produs**)malloc((sizeof(produs*))*n);
        int nr = 0;
        conversieListaVector(cap, vect, &nr);
        for (int i = 0; i < nr;i++)
        {
                printf("\n Cod= %d, denumire=%s, pret=%5.2f,
cantittate=%5.2f \n", *(vect[i]->cod), vect[i]-
>denumire, vect[i]->pret, vect[i]->cantitate);
                free(vect[i]->denumire);
        free(vect[i]->cod);
```

```c
    }

    free(vect);
    char denumire[20];
    printf("Denumire de sters ");
    scanf("%s", denumire);
    stergereNodDenumire(&cap, &coada, denumire);
    traversare(cap);
    traversareInversa(coada);
    dezalocare(cap);

}
```