```c
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "stdlib.h"
#include "malloc.h"
#include "string.h"

typedef struct Telefon
{
    float pret;
    char* marca;
}Telefon;

Telefon creareTelefon(float p, const char* m)
{
    Telefon t;
    t.pret = p;
    t.marca = (char*)malloc(sizeof(char) * (strlen(m) + 1));
    strcpy(t.marca, m);
    return t;
}

void afisareTelefon(Telefon t)
{
    printf("\n Telefonul cu marca %s are pretul %5.2f", t.marca, t.pret);
}

typedef struct Node
{
    Telefon info;
    struct Node* next;
}Node;

Node* creareNode(Node* next, Telefon t)
{
    Node* nou = (Node*)malloc(sizeof(Node));
    nou->info = creareTelefon(t.pret, t.marca);
    nou->next = next;
    return nou;
}

void afisareNode(Node* head)
{
    if (head)
    {
        Node* aux = head;
        while (aux) {
            afisareTelefon(aux->info);
            aux = aux->next;
        }
    }
}


//STACK

//inserare inceput
Node* pushStack(Node* head, Telefon t)
{
    Node* nou = creareNode(NULL, t);
    if (head)
    {
        nou->next = head;
        head = nou;
    }
    else {
        head = nou;
    }

    return head;
}

//extragere inceput

Telefon popStack(Node** head)
{
    if (*head) {
        Node* aux = *head;
        Telefon rezultat = creareTelefon(aux->info.pret, aux->info.marca);
        *head = (*head)->next; //se modifica direct in main, in timp real
        free(aux->info.marca);
        free(aux);
        return rezultat;
    }
    else {
        //returnam info redundanta
        return creareTelefon(-1, "");
    }
}


//inserare final
Node* pushStack2(Node* head, Telefon t)
{
    Node* nou = creareNode(NULL, t);
    if (head) {
        Node* aux = head;
        while (aux->next) {
            aux = aux->next;
        }
        aux->next = nou;
    }
    else {
        head = nou;
    }
    return head;
}

//extragere final
Telefon popStack2(Node** head)
{
    if (*head) {
        Node* aux = *head;
        //stergem ultimul nod
        //ne oprim pe penultimul
        while (aux->next->next) {
            aux = aux->next;
        }
        Telefon rezultate = creareTelefon(aux->next->info.pret, aux->next-
          >info.marca);
        free(aux->next->info.marca);
        free(aux->next);
        aux->next = NULL;
        return rezultate;
    }
    else {
        return creareTelefon(-1, "");
    }

}

//QUEUE

//inserare final
Node* pushQueue(Node* head, Telefon t)
{
    return pushStack2(head, t);
}

//extragere inceput
Telefon popQueue(Node** head) {
    return popStack(head);
}

//inserare inceput
Node* pushQueue2(Node* head, Telefon t) {
    return pushStack(head, t);
}
```

```c

//extragere final
Telefon popQueue2(Node** head) {
    return popStack2(head);
}

//HW 1 - inserare inainte unui nod cu un anumit pret
Node* inserareInainteaNodului(Node* head, Telefon t, float price)
{
    Node* nou = creareNode(NULL, t); //creat nod nou cu telefonul primit ca parametru
    if (head) { //daca exista stiva
        if (head->info.pret == price) {
            nou->next = head;
            head = nou;
        }
        else {
            Node* p = head; //ne luam un auxiliar
            while (p->next && p->next->info.pret != price) { //cat timp urmatorul nod are pretul diferit de cel
                //primit ca parametru, ma mut pe urmatorul nod
                p = p->next;
            }
            if (p->next->info.pret == price) // verificam din nou conditia
            {
                //refacem legaturile
                nou->next = p->next;
                p->next = nou;
            }
        }
    }
    else {
        head = nou;
    }
    return head;
}

//HW 2 - inserare dupa un nod cu un anumit pret
Node* inserareDupaUnNod(Node* head, Telefon t, float price)
{
    Node* nou = creareNode(NULL, t);
    if (head) {
        Node* p = head;
        while (p->next && p->info.pret != price) {
            p = p->next;
        }
        if (p->info.pret == price) {
            nou->next = p->next;
            p->next = nou;
        }
    }
    return head;
}

//HW 3 - extragere  nod cu un anumit pret
Telefon extragereNod(Node** head, float price)
{
    if (*head) {
        Node* p = *head;
        if ((*head)->info.pret == price) {
            Telefon rezultat = creareTelefon((*head)->info.pret, (*head)-
              >info.marca);
            Node* aux = (*head);
            (*head) = (*head)->next;
            free(aux->info.marca);
            free(aux);
            return rezultat;
        }
        else {


            while (p->next && p->next->info.pret != price) {
                p = p->next;
            }
            if (p->next) {
                Telefon rezultat = creareTelefon(p->next->info.pret, p->next-
                  >info.marca);
                Node* aux = p->next;
                p->next = p->next->next;
                free(aux->next->info.marca);
                free(aux);

                return rezultat;
            }
            else {
                return creareTelefon(-1, "");
            }
        }

    }
    else {
        return creareTelefon(-1, "");
    }

}

//HW 4 - stack 2 queue
Node* Stack2Queue(Node* head)
```

```c
{
    Node* queue = NULL;
    while (head) {
        //1. scot din stack
        //Telefon tmp = popStack(&head);

        //2. pus in queue
        //queue = pushQueue(queue, tmp);

        //same thing but in one line
        queue = pushQueue(queue, popStack(&head));
    }
}

//HW 5 - Queue to Stack

Node* Queue2Stack(Node* head)
{
    Node* stack = NULL;
    while (head)
    {
        stack = pushStack(stack, popQueue(&head));
    }
}
void main() {
    Node* stack = NULL;
    stack = pushStack(stack, creareTelefon(6500, "Apple"));
    stack = pushStack(stack, creareTelefon(4500, "Samsung"));
    stack = pushStack(stack, creareTelefon(4200, "Huawei"));
    stack = pushStack(stack, creareTelefon(1500, "Nokia"));
    stack = pushStack(stack, creareTelefon(8700, "Apple"));

    afisareNode(stack);


    Telefon t = popStack(&stack);
    printf("\n Afisare element extras ");
    afisareTelefon(t);
    //afisareTelefon(popStack(&stack));

    printf("\n Afisare lista dupa extragere ");
    afisareNode(stack);

    printf("\n Afisare element extras ");
    //afisareTelefon(t);
    afisareTelefon(popStack2(&stack));

    printf("\n Afisare lista dupa extragere ");
    afisareNode(stack);

    Telefon t1 = creareTelefon(100000, "GGHDFFDHVHD");
    printf("\n afisare dupa inserare inainte");
    stack = inserareInainteaNodului(stack, t1, 4200);
    afisareNode(stack);



    Telefon t2 = creareTelefon(5869, "GIGI");
    printf("\n afisare dupa inserare dupa un nod");
    stack = inserareDupaUnNod(stack, t2, 4500);
    afisareNode(stack);


    printf("\n Afisare element extras ");
    //afisareTelefon(t);
    afisareTelefon(extragereNod(&stack, 1500));
    printf("\nAfisare lista dupa extragere mijloc");
    afisareNode(stack);

    printf("\n Stack -> queue");
    Node* queue = Stack2Queue(stack);
    afisareNode(queue);

    printf("\n Queue -> Stack");
    Node* stack2 = Queue2Stack(queue);
    afisareNode(stack2);

}


```