```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#define TAMANHO 50

typedef struct {
    int dados[TAMANHO];
    int topo;
} Pilha;

void inicializar(Pilha *p){
    p->topo = -1;
}

bool isEmpty(Pilha *p){
    return p->topo == -1;
}

bool isFull(Pilha *p){
    return p->topo == TAMANHO - 1;
}

void push(Pilha *p, int valor){
    if(isFull(p)){
        printf("Pilha cheia\n");
        return;
    }
    p->topo++;
    p->dados[p->topo] = valor;
}
```

```
int pop(Pilha *p){
    if(isEmpty(p)){
        printf("Pilha vazia\n");
        return -1;
    }
    return p->dados[p->topo--];
}

int topo(Pilha *p){
    if(isEmpty(p)){
        printf("Pilha vazia\n");
        return -1;
    }
    return p->dados[p->topo];
}

void imprimir(Pilha *p){
    for(int i = 0; i <= p->topo; i++){
        printf("%d -> ", p->dados[i]);
    }
    printf("topo\n");
}

// 1. Inverter vetor
void inverterVetor(int v[], int tamanho){
    Pilha p;
    inicializar(&p);
    for(int i = 0; i < tamanho; i++){
        push(&p, v[i]);
    }
    for(int i = 0; i < tamanho; i++){
```

```
        v[i] = pop(&p);
    }
}

// 2. Verificar palíndromo
int ehPalindromo(char palavra[]){
    Pilha p;
    inicializar(&p);
    int len = strlen(palavra);
    for(int i = 0; i < len; i++){
        push(&p, palavra[i]);
    }
    for(int i = 0; i < len; i++){
        if(palavra[i] != pop(&p)){
            return 0;
        }
    }
    return 1;
}

// 3. Simular undo/redo
void simularUndoRedo(){
    Pilha desfazer, refazer;
    inicializar(&desfazer);
    inicializar(&refazer);

    push(&desfazer, 1);
    push(&desfazer, 2);
    push(&desfazer, 3);

    push(&refazer, pop(&desfazer));
    push(&desfazer, pop(&refazer));
```

```
    imprimir(&desfazer);
}

// 4. Remover pares
void removerPares(Pilha *orig){
    Pilha aux;
    inicializar(&aux);
    while(!isEmpty(orig)){
        int x = pop(orig);
        if(x % 2 != 0){
            push(&aux, x);
        }
    }
    while(!isEmpty(&aux)){
        push(orig, pop(&aux));
    }
}

// 5. Ordenar pilha
void ordenarPilha(Pilha *p){
    Pilha aux;
    inicializar(&aux);
    while(!isEmpty(p)){
        int tmp = pop(p);
        while(!isEmpty(&aux) && topo(&aux) > tmp){
            push(p, pop(&aux));
        }
        push(&aux, tmp);
    }
    while(!isEmpty(&aux)){
        push(p, pop(&aux));
```

```
    }
}

// MAIN TESTANDO TUDO
int main(){
    // 1. Inverter vetor
    int v[5] = {1, 2, 3, 4, 5};
    inverterVetor(v, 5);
    printf("Vetor invertido: ");
    for(int i = 0; i < 5; i++){
        printf("%d ", v[i]);
    }
    printf("\n");

    // 2. Palíndromo
    char palavra[30] = "osso";
    if(ehPalindromo(palavra)){
        printf("É palíndromo\n");
    } else {
        printf("Não é palíndromo\n");
    }

    // 3. Undo/Redo
    printf("Simulando Undo/Redo:\n");
    simularUndoRedo();

    // 4. Remover pares
    Pilha p1;
    inicializar(&p1);
    push(&p1, 1);
    push(&p1, 2);
    push(&p1, 3);
```

```
    push(&p1, 4);
    push(&p1, 5);
    removerPares(&p1);
    printf("Pilha sem pares: ");
    imprimir(&p1);

    // 5. Ordenar pilha
    Pilha p2;
    inicializar(&p2);
    push(&p2, 3);
    push(&p2, 1);
    push(&p2, 4);
    push(&p2, 2);
    ordenarPilha(&p2);
    printf("Pilha ordenada: ");
    imprimir(&p2);

    return 0;
}
```