pilha.h

```
/* TAD: Pilha */
/* Tipo Exportado */

typedef struct pilha Pilha;

/* criar uma estrutura de pilha */
Pilha* pilha_cria (void);
/* inserir um elemento no topo (push) */
void pilha_push (Pilha* p, float v);
/* remover o elemento do topo (pop) */
float pilha_pop (Pilha* p);
/* verificar se a pilha está vazia */
int pilha_vazia (Pilha* p);
/* liberar a estrutura de pilha */
void pilha_libera (Pilha* p);

/* imprime: versão com lista */
void pilha_imprime (Pilha* p);
```

pilha_code.c

```
#include <stdio.h>
#include "pilha.h"
int main (void) {
int i, elem;
float conteudo;
Pilha* l1;
Pilha* l2;
l1 = pilha_cria();
l2 = pilha_cria();
printf("\n Digite o numero de elementos da pilha1: ");
scanf("%d", &elem);
for (i=1 ; i <= elem ; i++){
 printf("\n Digite o elemento %d da pilha1: ",i);
 scanf("%f", &conteudo);
 pilha_push (l1, conteudo);
}
printf("\n Digite o numero de elementos da pilha2: ");
scanf("%d", &elem);
for (i=1 ; i <= elem ; i++){
 printf("\n Digite o elemento %d da pilha2: ",i);
 scanf("%f", &conteudo);
 pilha_push (l2, conteudo);
}
printf("\n Imprimindo pilha1\n"); pilha_imprime(l1);
printf("\n Imprimindo pilha2\n"); pilha_imprime(l2);
printf("\n Concatenando pilha1 e pilha 2  \n");
pilha_concatena(l1,l2);
printf("\n Imprimindo pilha concatenada \n");
pilha_imprime(l1);
pilha_libera(l1);
pilha_libera(l2);
return 0;
}
```

Implemente uma função que receba duas pilhas, p1 e p2, e passe todos os elementos da pilha p2 para o topo de p1. Note que ao final dessa função, a pilha p2 vai estar vazia, e a pilha p1 conterá todos os elementos das 2 pilhas. Essa função deve obedecer ao protótipo:

void pilha_concatena (Pilha* p1, Pilha* p2);

pilha.c

```
#include <stdio.h>
#include "pilha.h"
struct no {
float info;
struct no* prox;
};
typedef struct no No;
struct pilha {
No* prim;
};
Pilha* pilha_cria (void) {
  Pilha* p = (Pilha*) malloc(sizeof(Pilha));
  p->prim = NULL;
  return p;
}
/* função auxiliar: insere no início */
No* ins_ini (No* l, float v) {
   No* p = (No*) malloc(sizeof(No));
   p->info = v;
   p->prox = l;
   return p;
}
/* função auxiliar: retira do início */
No* ret_ini (No* l) {
    No* p = l->prox;
    free(l);
    return p;
}
/* inserir um elemento no topo (push) */
void pilha_push (Pilha* p, float v) {
   p->prim = ins_ini(p->prim,v);
}
/* remover o elemento do topo (pop) */
float pilha_pop (Pilha* p) {
   float v;
   if (pilha_vazia(p)) {
      printf("Pilha vazia.\n");
      exit(1); /* aborta programa */
    }
    v = p->prim->info;
    p->prim = ret_ini(p->prim);
    return v;
}
int pilha_vazia (Pilha* p) {/* verificar se a pilha está vazia */
   return (p->prim==NULL);
}
void pilha_libera (Pilha* p) {/* liberar a estrutura de pilha */
   No* q = p->prim;
   while (q!=NULL) {
       No* t = q->prox;
       free(q);
       q = t;
    }
    free(p);
}
void pilha_imprime (Pilha* p){  /* imprime: versão  lista */
   No* q;
   for (q=p->prim; q!=NULL; q=q->prox)
   printf("%f\n",q->info);
}
```