Nomes:
Henrique Sartori Siqueira **19240472**
Rafael SIlva Barbon **19243633**

Obs. Os exercícios foram realizados usando a função vazia retornando 0 se tiver vazia e 1 se não estiver vazia. (Utilizando a primeira biblioteca que a professora Lúcia disponibilizou) .

EX1:

```
#include <stdio.h>
#include <stdlib.h>
#include "PILHA.h"

Pilha* inverte(Pilha *p);
Pilha* intersecao(Pilha **p1,Pilha **p2);

int main()
{
   Pilha *p1 = CriaPilha(), *p2 = CriaPilha(),*p3;
   int info;
   do{

      printf("\n\tInsira um valor para a pilha 1 (-999 para sair): ");
      scanf("%d",&info);
      if(info != -999)
         push(p1,info);
   }while(info != -999);
   do{

      printf("\n\tInsira um valor para a pilha 2 (-999 para sair): ");
      scanf("%d",&info);
      if(info != -999)
         push(p2,info);
   }while(info != -999);

   p3 = intersecao(&p1,&p2);

   printf("\n\n\n");
   imprime(p1);
   printf("\n\n\n");
   imprime(p2);
   printf("\n\n\n");
   imprime(p3);

   p1 = libera(p1);
```

```
    p2 = libera(p2);
    p3 = libera(p3);
    return 0;
}

Pilha* inverte(Pilha *p){
        Pilha *aux = CriaPilha();
        int x;
        while(vazia(p)){
                x = pop(p);
                push(aux,x);
        }
        return aux;
}

Pilha* intersecao(Pilha **p1,Pilha **p2){
    Pilha *p1inv = CriaPilha(),*p2inv = CriaPilha(), *inter = CriaPilha();
    int val,conf;

    while(vazia(*p1)){
        val = pop(*p1);
        while(vazia(*p2)){
            conf = pop(*p2);
            if(val == conf)
                push(inter,val);
            push(p2inv,conf);
        }
        push(p1inv,val);
        *p2 = inverte(p2inv);
    }
    *p1 = inverte(p1inv);
    p1inv = libera(p1inv);
    p2inv = libera(p2inv);
        inter = inverte(inter);
    return inter;
}
```

EX2:

```
/*

Em termos computacionais de desempenho haveria menos loops, pois so haveria a
necessidade de
fazer a conferencia das pilhas somente uma vez para cada valor e inverte-la ao final.
Porem, em termos de codigo,
havera mais complexidade dentro da funcao, ou seja, tera condicionais de verificacao para
ambas as pilhas a fim de que
```

os valores se atualizem, igualando-se para que haja a interseccao.

*/

```
#include <stdio.h>
#include <stdlib.h>
#include "PILHA.h"

Pilha* inverte(Pilha *p);
void volta(Pilha ***p,Pilha *inv);
Pilha* intersecao(Pilha **p1,Pilha **p2);

int main()
{
   Pilha *p1 = CriaPilha(), *p2 = CriaPilha(),*p3;
   int info;
   do{

      printf("\n\tInsira um valor para a pilha 1 (-999 para sair): ");
      scanf("%d",&info);
      if(info != -999)
         push(p1,info);
   }while(info != -999);
   do{

      printf("\n\tInsira um valor para a pilha 2 (-999 para sair): ");
      scanf("%d",&info);
      if(info != -999)
         push(p2,info);
   }while(info != -999);

   p3 = intersecao(&p1,&p2);

   printf("\n\n\n");
   imprime(p1);
   printf("\n\n\n");
   imprime(p2);
   printf("\n\n\n");
   imprime(p3);

   p1 = libera(p1);
   p2 = libera(p2);
```

```
    p3 = libera(p3);
    return 0;
}

Pilha* inverte(Pilha *p){
        Pilha *aux = CriaPilha();
        int x;
        while(vazia(p)){
                x = pop(p);
                push(aux,x);
        }
        return aux;
}

void volta(Pilha ***p,Pilha *inv){
        //Pilha *aux = CriaPilha();
        int x;
        while(vazia(inv)){
                x = pop(inv);
                push(**p,x);
        }
}

Pilha* intersecao(Pilha **p1,Pilha **p2){
    Pilha *p1inv = CriaPilha(),*p2inv = CriaPilha(), *inter = CriaPilha();
    int val1,val2;
        val1 = pop(*p1);
        val2 = pop(*p2);
    while(1){
            if(val1 == val2){
                push(inter,val1);
        push(p2inv,val2);
                push(p1inv,val1);
                        if(vazia(*p1)&&vazia(*p2)){
                                val1 = pop(*p1);
                                val2 = pop(*p2);
                        }
                        else
                                break;
                }else if(val1 > val2){
                        push(p1inv,val1);
                        if(vazia(*p1))
                                val1 = pop(*p1);
                        else
                                break;
                }else{
```

```
                        push(p2inv,val2);
                        if(vazia(*p2))
                                val2 = pop(*p2);
                        else
                                break;
                }
    }
        volta(&p2,p2inv);
    volta(&p1,p1inv);
    p1inv = libera(p1inv);
    p2inv = libera(p2inv);
        inter = inverte(inter);
    return inter;
}
```

EX3:

```
#include <stdio.h>
#include <stdlib.h>
#include "PILHA.h"

Pilha* menor_elemento(Pilha *p,int *menor);
Pilha* maior_elemento(Pilha *p,int *maior);

int main()
{
    int info,maior,menor;
    Pilha *p=CriaPilha();
    do{
        printf("\nInsira os elementos da lista(-999 para sair):");
        scanf("%d",&info);
        if(info!=-999)
            push(p,info);
    }while(info!=-999);
    p=maior_elemento(p,&maior);
    p=menor_elemento(p,&menor);
    imprime(p);
    printf("\n\n\tMaior elemento:%d\n",maior);
    printf("\n\tMenor elemento:%d\n",menor);
    p=libera(p);
    return 0;
}

Pilha* maior_elemento(Pilha *p,int *maior)
{
    int valor;
    Pilha *aux=CriaPilha();
```

```
    *maior=pop(p);
    push(aux,*maior);
    while(vazia(p))
    {
        valor=pop(p);
        if(valor>*maior)
            *maior=valor;
        push(aux,valor);
    }
    return aux;
}

Pilha* menor_elemento(Pilha *p,int *menor)
{
    int valor;
    Pilha *aux=CriaPilha();
    *menor=pop(p);
    push(aux,*menor);
    while(vazia(p))
    {
        valor=pop(p);
        if(valor<*menor)
            *menor=valor;
        push(aux,valor);
    }
    return aux;
}
///Nao foi necessario inverter a pilha nas funções, como a função inverte as pilhas
// e são chamadas duas vezes a pilha acaba voltando para a sequencia original;
```

EX4:

```
#include <stdio.h>
#include <stdlib.h>
#include "PILHA.h"

Pilha* ordena(Pilha *p,int tam);
Pilha* tira_maior_pilha(Pilha *p,int maior);

int main()
{
    Pilha *p = CriaPilha();
    int info,cont=0;
    do{
        printf("\n\tInsira um valor para a pilha (-111 para sair): ");
```

```
        scanf("%d",&info);
        if(info != -111)
        {
            push(p,info);
            cont++;
        }
    }while(info != -111);
    printf("\n\tPilha Original:");
    imprime(p);
    p = ordena(p,cont);
    printf("\n\n\tPilha Ordenada:");
    imprime(p);
    printf("\n");
    p = libera(p);

    return 0;
}

Pilha* ordena(Pilha *p,int tam)
{
    Pilha *aux,*ordenada=CriaPilha();
    int i,maior,valor;
    for(i=0;i<tam;i++)
    {
        Pilha *aux = CriaPilha();
        maior=pop(p);///tira da pilha
        push(p,maior);///devolve para pilha
        while(vazia(p))
        {
            valor=pop(p);
            if(valor>maior)
                maior=valor;
            push(aux,valor);
        }
        push(ordenada,maior);
        p=tira_maior_pilha(aux,maior);///vai tirar o maior elemento da pilha que foi adicionado
na ordenada, para continuar a comparação com os menores.
        libera(aux);
    }
    return ordenada;
}

Pilha* tira_maior_pilha(Pilha *p,int maior)
{
    Pilha *aux=CriaPilha();
    int valor;
```

```
    while(vazia(p))
    {
        valor=pop(p);
        if(valor!=maior)
            push(aux,valor);
    }
    return aux;
}
```