1ª) Implemente o Tipo Abstrato de Dados (TAD) "lista.h" (ver slides sobre Listas Encadeadas) e acrescente as seguintes funções:

a) função para retornar o número de nós da lista que possuem o campo `info` com valor menor que `n`. Essa função deve obedecer ao protótipo:

```
int menores(Lista* l, int n);
```

b) função para somar os valores do campo `info` de todos os nós. Essa função deve obedecer ao protótipo:

```
int soma(Lista* l);
```

c) função para retornar o número de nós da lista que possuem o campo `info` com `n` divisores positivos. Essa função deve obedecer ao protótipo:

```
int num_ndivp(Lista* l, int n);
```

d) função para gerar uma **nova** lista que é a intersecção de duas listas. Os valores na nova lista devem estar ordenados.

Essa função deve obedecer ao protótipo:

```
Lista* lst_intersec(Lista* l1, Lista* l2);
```

Por exemplo, se lista $L_1 \rightarrow 5 \rightarrow 7 \rightarrow 2 \rightarrow 4 \rightarrow //$ e lista $L_2 \rightarrow 7 \rightarrow 9 \rightarrow 2 \rightarrow //$, a chamada `Lista* L3= lst_intersec(L1,L2)` gera a nova lista $L_3 \rightarrow 2 \rightarrow 7 \rightarrow //$.

e) função para gerar uma **nova** lista que é a concatenação de uma lista `l1` no final de uma lista `l2`. Essa função deve obedecer ao protótipo:

```
Lista* lst_conc(Lista* l1, Lista* l2);
```

f) função que faça a diferença de duas listas `l1` e `l2` (ou seja, que retire de `l1` os elementos que estão em `l2`). Essa função deve obedecer ao protótipo:

```
Lista* lst_diferenca(Lista* l1, Lista* l2);
```

Por exemplo, se lista $L_1 \rightarrow 3 \rightarrow 7 \rightarrow 2 \rightarrow 4 \rightarrow //$ e lista $L_2 \rightarrow 7 \rightarrow 9 \rightarrow //$, a chamada `Lista* L1 = lst_diferenca(L1, L2)` altera a primeira lista para $L_1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow //$.

A seguir, execute o seguinte programa.

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

int main(void){

  int a, b, c;
  Lista* l1 = lst_cria();
  l1 = lst_insere(l1,6);
  l1 = lst_insere(l1,12);
  l1 = lst_insere(l1,25);
  l1 = lst_insere(l1,28);
  l1 = lst_insere(l1,45);
  l1 = lst_remove(l1,25);
  l1 = lst_remove_rec(l1,12);
  lst_imprime(l1);
  lst_imprime_invertida_rec(l1);

  a=soma(l1); b=menores(l1,22); c=num_ndivp(l1,6);
  printf("Soma dos valores dos nós %d\n",a);
  printf("Num. nós c/ info < que 22: %d\n",b);
  printf("Num. nós c/ info c/ 6 div. positivos.: %d\n",c);

  Lista* l2 = lst_cria();
  l2 = lst_insere(l2,28);
  l2 = lst_insere(l2,130);
  l2 = lst_insere(l2,6);

  Lista* l3=lst_conc(l1,l2);
  lst_imprime(l3);

  Lista* l4=lst_intersec(l1,l2);
  lst_imprime(l4);

  l1=lst_diferenca(l1,l2);
  lst_imprime(l1);

  lst_libera(l1);
  lst_libera(l2);
  lst_libera(l3);
  lst_libera(l4);

  system("PAUSE");
  return 0;
}
```

2ª) Implemente o Tipo Abstrato de Dados (TAD) "pilha.h" usando Listas Encadeadas (ver slides sobre Pilhas) e acrescente as seguintes funções:

a) função para gerar uma nova pilha com os elementos da pilha `p` na ordem inversa. Essa função deve obedecer ao protótipo:

```
Pilha* inverte_pilha(Pilha* p);
```

b) função que verifique quais são os elementos em comum em duas listas `l1` e `l2` e que os empilhe em **ordem crescente** em uma nova pilha. Essa função deve obedecer ao protótipo:

```
Pilha* empilha_elem_comuns(Lista* l1, Lista* l2);
```

A seguir, execute o seguinte programa.

```
#include <stdio.h>
#include <stdlib.h>
#include "lista.h"
#include "pilha.h"

int main(void){
  int a;
  Pilha* p1 = pilha_cria();

  pilha_push(p1,10);
  pilha_push(p1,20);
  pilha_push(p1,25);
  pilha_push(p1,30);
  a = pilha_pop(p1);
  printf("Elemento removido da pilha p1: %d\n",a);

  Lista* l1=lst_cria();
  l1=lst_insere(l1,4);
  l1=lst_insere(l1,5);
  l1=lst_insere(l1,6);
  l1=lst_insere(l1,7);

  Lista* l2=lst_cria();
  l2=lst_insere(l2,5);
  l2=lst_insere(l2,6);
  l2=lst_insere(l2,7);
  l2=lst_insere(l2,8);

  Pilha* p2= empilha_elem_comuns(l1,l2);
  pilha_imprime(p2);

  Pilha* p3= inverte_pilha(p2);
  pilha_imprime(p3);
```

```
  lst_libera(l1); lst_libera(l2);
  pilha_libera(p1); pilha_libera(p2); pilha_libera(p3);

  system("PAUSE");
  return 0;
}
```

3ª) Implemente o Tipo Abstrato de Dados (TAD) "fila1.h"(implementação com vetor) e "fila2.h" (implementação com listas encadeadas) e acrescente as seguintes funções:

a) função para retornar o número de elementos da fila com valor primo. Essa função deve obedecer ao protótipo:

```
int qtd_primos(Fila* f, int n);
```

b) função que crie uma **nova** fila com os elementos da fila `f` na ordem inversa. Essa função deve obedecer ao protótipo:

```
Fila* inverte_fila(Fila* f);
```

A seguir, execute o seguinte programa com as TAD's "fila1.h" e "fila2.h"

```
#include <stdio.h>
#include<stdlib.h>
#include "fila1.h" //executar também com "fila2.h"

int main(void){
  int a, qtd;
  Fila* f1 = fila_cria();
  fila_insere(f1,31);
  fila_insere(f1,33);
  fila_insere(f1,35);
  fila_insere(f1,37);
  fila_insere(f1,39);
  a = fila_remove(f1);
  printf(''Valor removido da fila f1: %d\n'',a);
  fila_imprime(f1);

  Fila* f2=inverte_fila(f1);
  fila_imprime(f2);

  qtd=qtd_primos(f1);
  printf(''Qtd. elem. primos na fila f1: %d\n'',qtd);

  fila_libera(f1); fila_libera(f2);

  system("PAUSE");
  return 0;
}
```