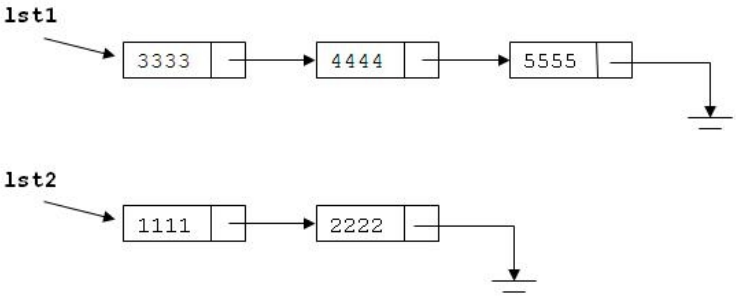
Escreva uma função que receba como parâmetros duas listas encadeadas, **lst1** e **lst2**, e usando uma pilha, retorne uma terceira lista, que seja o resultado da concatenação de **lst2** com **lst1**.

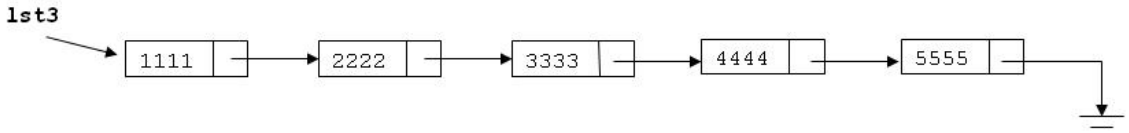
A figura a seguir explica o objetivo da questão:

Os nós das listas envolvidas no problema são definidos pela estrutura a seguir:

```
struct elemento
```



Resultado da Concatenação




```
{
      int info;
      struct elemento *prox;
};
typedef struct elemento Elemento;
```

Essa função deve obedecer ao seguinte protótipo:

```
Elemento* lst_concatena(Elemento* lst1,Elemento* lst2);
```

Considere que as seguintes funções dos tipos abstratos de dados Pilha e Lista estejam disponíveis:

Pilha* pilha_cria (void);	Retorna o ponteiro para uma nova pilha alocada dinamicamente.
int pilha_pop (Pilha* p);	Retira um elemento do topo de uma pilha. O ponteiro da pilha é passado como parâmetro, e o retorno é o valor deste elemento.
void pilha_push (Pilha* p, int x);	Insere um elemento no topo de uma pilha. O

	ponteiro da pilha e o elemento a ser inserido são passados como parâmetros.
int pilha_vazia (Pilha* p);	Verifica se a pilha está vazia. O ponteiro da pilha é passado como parâmetro e o retorno é 1, se a pilha estiver vazia, ou 0, caso contrário.
void pilha_libera (Pilha* p);	Esvazia e libera a memória alocada para uma pilha. O ponteiro da pilha é passado como parâmetro.

Elemento* lst_cria(void);	Retorna o ponteiro para uma nova lista alocada dinamicamente.
Elemento* lst_insere(Elemento* lst, int i);	Insere um novo elemento no início de uma lista. O ponteiro para a lista e o elemento a ser inserido são passados como parâmetros.
int lst_vazia(Elemento* lst);	Retorna 1 se a lista estiver vazia, ou 0, caso contrario.
int lst_libera(Elemento* lst);	Libera toda a memória alocada para uma lista. O ponteiro para a lista é passado como parâmetro.