ATIVIDADE DE LABORATÓRIO 01
CONJUNTOS COM LISTAS LIGADAS

A Teoria dos Conjuntos pode ser utilizada na definição de muitos elementos matemáticos. Um conjunto é uma coleção de elementos não ordenados, e com base nisso muitos conceitos podem ser definidos. Você deve, com o auxílio de listas ligadas, implementar as operações básicas de conjuntos de números naturais. As listas **devem possuir nó cabeça** e seus elementos **devem permanecer em ordem crescente**.

Sua implementação deve possuir os seguintes tipos abstratos de dados (TADs)

```
struct lista_encadeada{
    int elemento;
    struct lista_encadeada* proximo_elemento;
};

struct conjunto{
    lista_encadeada* lista;
};
```

Você deve implementar as funções abaixo, que devem operar sobre os TADs descritos acima.

```
/*Cria um conjunto vazio, alocando memória para um novo conjunto*/
struct conjunto* criar_conjunto();

/*Desaloca a memória alocada para o conjunto c*/
void deletar_conjunto( struct conjunto* c );

/*Insere o elemento e no conjunto c, de modo que a lista de c permaneca ordenada.
OBS1: a ordenacao é para manter as saídas iguais aos do gabarito do susy
OBS2: os conjuntos não devem conter elementos repetidos*/
void inserir_elemento( struct conjunto* c, int e );

/*Remove o elemento e da lista encadeada do conjunto c.*/
void remover_elemento( struct conjunto* c, int e );

/*Retorna 1 se o elemento e pertence ao conjunto c, ou 0, caso contrário*/
int pertinencia( struct conjunto* c, int e );

/*Retorna um novo conjunto contendo os elementos comuns aos conjuntos c1 e c2.*/
struct conjunto* intersecao( struct conjunto* c1, struct conjunto* c2 );

/*Retorna um novo conjunto contendo a união dos elementos dos conjuntos c1 e c2.*/
struct conjunto* uniao( struct conjunto* c1, struct conjunto* c2 );

/*Retorna um novo conjunto contendo a diferenca dos conjuntos c1 e c2.*/
struct conjunto* diferenca( struct conjunto* c1, struct conjunto* c2 );

/*Retorna 1 se c1 é subconjunto de c2, ou 0 caso contrário.*/
int subconjunto( struct conjunto* c1, struct conjunto* c2 );

/*Retorna 1 se o conjunto c1 é igual ao conjunto c2, ou 0 caso contrário.*/
int igualdade( struct conjunto* c1, struct conjunto* c2 );
```

```
/*Retorna a cardinalidade do conjunto c*/
int cardinalidade( struct conjunto* c );

/*Imprime na tela os elementos presentes na lista encadeada do conjunto c. Você não deve modificar esta função*/
void listar_elementos( struct conjunto* c ){
	if( !c ) return;
	struct lista_encadeada* pt = c->lista;
	printf("{");
	while( pt->proximo_elemento ){
		printf(" %d", pt->proximo_elemento->elemento );
		pt = pt->proximo_elemento;
		if( pt->proximo_elemento )
			printf(",");
	}
	printf(" }\n");
}
```

Você deve criar um conjunto com dez elementos do tipo *struct conjunto** ou do tipo *struct conjunto* e inicializá-los como conjuntos vazios. Após a inicialização, para testar a sua implementação das funções, o seu programa deve ler várias linhas de casos de teste. Cada linha de teste começa com um caracter que define a operação a ser realizada, seguido de parâmetros que variam de acordo com a operação.

Abaixo estão as definições das operações, seus respectivos caracteres e parâmetros.

Caracter: A
Parâmetros: C E
Operação: Adicionar o inteiro E no conjunto indexado pelo inteiro C (1 <= C <= 10)

Caracter: R
Parâmetros: C E
Operação: Remove o inteiro E no conjunto indexado pelo inteiro C

Caracter: L
Parâmetros: C
Operação: Imprime na saída padrão os elementos do conjunto indexado pelo inteiro C, no padrão { E1, E2, …, En }. OBS: Use a função listar_elementos fornecida acima.

Caracter: F
Parâmetros: C
Operação: Destrói o conjunto indexado pelo inteiro C, substituindo-o por um conjunto vazio

Caracter: I
Parâmetros: C1 C2 C3
Operação: Calcula um novo conjunto definido pela interseção entre os conjuntos indexados pelos inteiros C1 e C2, e o armazena no índice determinado pelo inteiro C3. O conjunto que existia na posição C3 do vetor deve ser destruído e substituído pelo novo conjunto calculado.

Caracter: U
Parâmetros: C1 C2 C3
Operação: Calcula um novo conjunto definido pela união entre os conjuntos indexados pelos inteiros C1 e C2, e o armazena no índice determinado pelo inteiro C3. O conjunto que existia na posição C3 do vetor deve ser destruído e substituído pelo novo conjunto calculado.

Caracter: P
Parâmetros: C E
Operação: Imprime a string “S\n” na saída padrão se o inteiro E está presente na lista encadeada do conjunto indexado pelo inteiro C. Caso contrário imprime “N\n”.

Caracter: D
Parâmetros: C1 C2 C3
Operação: Calcula um novo conjunto definido pela diferença entre os conjuntos indexados pelos inteiros C1 e C2, e o armazena no índice determinado pelo inteiro C3. O conjunto que existia na posição C3 do vetor deve ser destruído e substituído pelo novo conjunto calculado.

Caracter: C
Parâmetros: C1 C2
Operação: Imprime a string “S\n” na saída padrão caso o conjunto indexado pelo inteiro C1 seja subconjunto do conjunto indexado pelo inteiro C2. Imprime “N\n” caso contrário
Exemplo:

Caracter: =
Parâmetros: C1 C2
Operação: Imprime a string “S\n” na saída padrão caso o conjunto indexado pelo inteiro C1 seja igual ao conjunto indexado pelo inteiro C2. Imprime “N\n” caso contrário

Caracter: #
Parâmetros: C
Operação: Imprime na saída padrão a cardinalidade do conjunto indexado pelo inteiro C

Caracter: X
Parâmetros: Nenhum
Operação: Termina a execução do programa.

Note que as operações como intersecção, união, igualdade e subconjunto devem ser implementadas utilizando intercalação (não percorra a lista ligada mais de uma vez).

Exemplo de entradas e saídas

Entrada	Saída
A 1 1	
A 1 2	
A 1 3	
A 2 3	
A 2 2	
A 2 1	
A 2 1	
L 1	{ 1, 2, 3 }
L 2	{ 1, 2, 3 }
R 2 3	
A 2 4	
L 2	{ 1, 2, 4 }
I 1 2 3	

U 1 2 4	
D 1 2 5	
L 3	{ 1, 2 }
L 4	{ 1, 2, 3, 4 }
L 5	{ 3 }
C 5 4	S
C 5 10	N
= 1 2	N
A 9 3	
= 9 5	S
P 4 1	S
# 4	4
F 4	
# 4	0
X	

OBS: Na sua implementação você deverá utilizar o pacote balloc do Professor Tomasz Kowaltowski para o gerenciamento da memória dinâmica. No cabeçalho dos arquivos deve haver a linha #include “balloc.h” e no lugar dos comandos malloc() e free(), devem ser usado os comandos (Macros) MALLOC() e FREE(). Assim, caso haja "vazamento" de memória o programa indicará onde foi feita a alocação que não foi liberada. O pacote (os arquivos balloc.c e balloc.h) estão na seção arquivos auxiliares, disponíveis no arquivo auxiliares.zip ou separadamente.

Obs.

- Serão permitidas no máximo 15 submissões
- O programa deve ser implementado em C
- O aluno pode modificar algumas coisas no arquivo main.c desde que sejam mudanças justificáveis
- O pacote balloc DEVE ser utilizado