ACH2023 – turma 03 – EP1 – 1º Semestre de 2022

Trabalho individual – entrega pelo sistema eDisciplinas até 30/10/2022

Gerenciamento de Turmas de Alunos

O objetivo deste EP é gerenciar uma estrutura aqui chamada de TURMA que conterá informações sobre alunos (com nusp, nota e frequência) pertencentes a essa estrutura.

A estrutura será composta por **duas listas ligadas ordenadas circulares e com nó-cabeça**. Estas listas serão listas de **referências aos registros dos alunos** e conterão os mesmos alunos, a diferença será a forma de ordenação. Em uma das listas os alunos estarão ordenados de acordo com seu número USP (nusp), já na outra a ordenação será de acordo com a nota.

Seu objetivo é implementar funções para gerenciar essa estrutura

Dentre as operações previstas para esta estrutura estão (em negrito estão destacadas as funções que deverão ser implementadas por você neste EP):

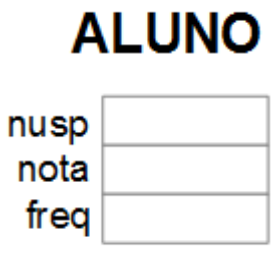
- criação/inicialização de uma turma;
- limpeza/reinicialização de uma turma;
- listagem de todos os alunos da turma;
- **consulta à quantidade de alunos na turma (tamanho);**
- **busca por um aluno na turma;**
- **inserção de um aluno na turma;**
- **exclusão de um aluno na turma.**

Para este EP, você deverá implementar um conjunto de funções de gerenciamento de listas utilizando principalmente os conceitos: **listas ligadas ordenadas circulares e com nó-cabeça**.

A seguir são apresentadas as estruturas de dados envolvidas nesta implementação e como elas serão gerenciadas.

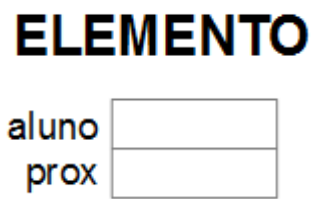
As informações dos alunos serão armazenadas na estrutura *ALUNO*, que contém três campos: *nusp* (número USP do aluno), *nota* (nota do aluno na turma) e *freq* (frequência do aluno na turma).

```
typedef struct {
     int nusp;
     int nota;
     int freq;
} ALUNO;
```



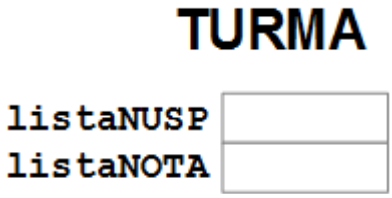
A estrutura básica das listas ligadas será o *ELEMENTO*, que contém dois campos: *aluno* (ponteiro/referência para um aluno) e *prox* (ponteiro para o endereço do elemento posterior ao atual).

```
typedef struct aux{
     ALUNO* aluno;
     struct aux* prox;
} ELEMENTO;

typedef ELEMENTO* PONT;
```



A estrutura *TURMA* possui dois campos: *listaNUSP* que é um ponteiro/referência para o nó-cabeça da lista ligada de elementos ordenada pelo número USP dos respectivos alunos (apontados em cada elemento); e listaNOTA que é um ponteiro/referência para o nó-cabeça da lista ligada de elementos ordenada pela nota dos respectivos alunos (apontados em cada elemento). **Cada lista de elementos será ordenada, circular e possuirá um nó-cabeça.**

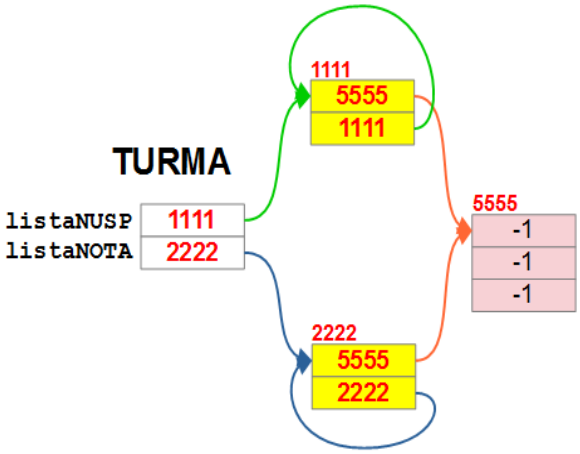
```
typedef struct {
     PONT listaNUSP;
     PONT listaNOTA;
} TURMA;
```



A função inicializaTurma é responsável por criar uma nova turma, incluindo a criação dos nós-cabeça para encabeçar cada uma das listas. **Estes nós-cabeça, bem como o aluno "fictício" apontado por eles, nunca deverão ser apagados/excluídos.**

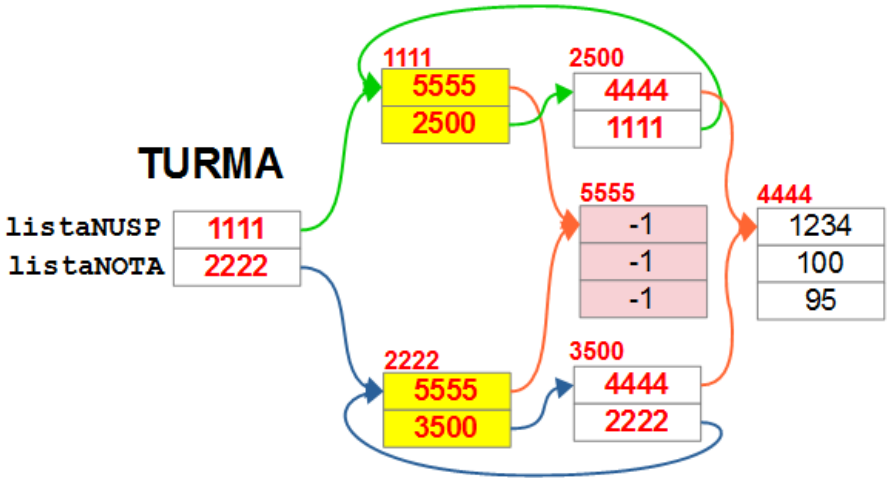
```
TURMA inicializaTurma(){
     TURMA t1;

     // criação de um aluno fictício que poder ser usado como sentinela
     ALUNO* ficticio = (ALUNO*) malloc(sizeof(ALUNO));
     ficticio->nusp = -1;
     ficticio->nota = -1;
     ficticio->freq = -1;


     // criação do primeiro nó-cabeça
     t1.listaNUSP = (PONT) malloc(sizeof(ELEMENTO));
     t1.listaNUSP->aluno = ficticio;
     t1.listaNUSP->prox = t1.listaNUSP;


     // criação do segundo nó-cabeça
     t1.listaNOTA = (PONT) malloc(sizeof(ELEMENTO));
     t1.listaNOTA->aluno = ficticio;
     t1.listaNOTA->prox = t1.listaNOTA;

     return t1;
}
```

Exemplo de turma recém-criada:

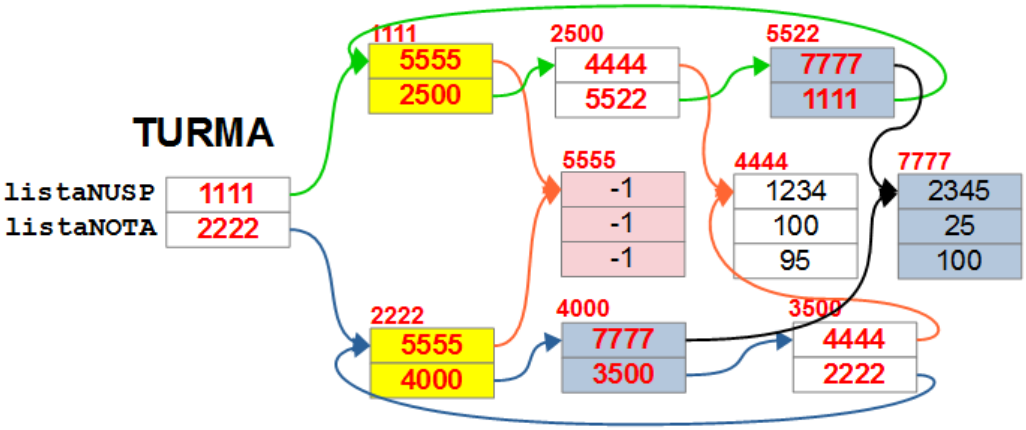


Ao se inserir o primeiro aluno na estrutura, este deverá ser incluído como próximo elemento do nó-cabeça, tanto da lista ordenada por número USP, quanto da lista ordenada por nota.

Exemplo de turma após a inserção do aluno com nusp=1234, nota=100 e freq=95



Exemplo da mesma lista após a inserção do aluno com nusp=2345, nota=25 e freq=100. Observe que na lista ordenada por número USP este elemento se tornou o último, já na lista ordenada por notas, ele ficou em primeiro lugar (logo após o nó-cabeça).



Funções que deverão ser implementadas no EP

int tamanho(TURMA turma)*: função que retorna o número de alunos da turma cujo endereço foi passado como parâmetro de entrada.

ALUNO buscarAluno(TURMA* turma, int nusp)*: função que recebe o endereço de uma turma e um número USP e retorna *NULL* caso não exista um aluno com o respectivo número USP na turma ou o endereço de memória do registro do respectivo aluno, caso ele esteja presente na turma.

bool inserirAluno(TURMA turma, int nusp, int nota, int frequencia)*: função que recebe o endereço de uma turma e o número USP, nota e frequência de um aluno e deve: Retornar false caso o número USP seja menor do que zero, ou a nota seja menor do que zero ou maior do que cem ou a frequência seja menor do que zero ou maior do que cem; Deve, também, retornar false se a turma já contiver um aluno com o mesmo número USP. Caso contrário, esse novo aluno deverá ser inserido na turma e a função deverá retornar *true*. A inserção na turma contém as seguintes atividades:
1a) alocação dinâmica de memória para uma estrutura do tipo ALUNO e preenchimento de seus campos com os respectivos parâmetros recebidos pela Função de inserção;
2a) alocação dinâmica de memória de duas estruturas do tipo ELEMENTO (ambas referenciarão o novo aluno por meio de seu campo *aluno*. Uma delas deverá ser inserida na posição correta da lista ordenada por *nusp* e a outra na posição correta da lista ordenada por nota.
3a) por fim, a Função deverá retornar *true*.

bool excluirAluno(TURMA turma, int nusp)*: função que recebe o endereço de uma turma e um número USP e deve: Retornar false se a turma não possuir um aluno com esse número USP. Caso contrário, esse aluno deverá ser excluído da turma e a Função deverá retornar *true*. A exclusão na turma contém as seguintes atividades:
1a) remoção dos elementos que apontam para o respectivo aluno das duas listas (ordenada por *nusp* e ordenada por nota).
2a) liberação da memória do registro do aluno e dos elementos das listas que apontavam para o respectivo aluno.
3a) por fim, a Função deverá retornar *true*.

Informações gerais:

Os EPs desta disciplina são trabalhos individuais que devem ser submetidos pelos alunos via sistema eDisciplinas (https://edisciplinas.usp.br/) até às 23:59h do dia 30/10/2022 (com margem de tolerância de 60 minutos).

Você receberá três arquivos para este EP:

- turma.h que contém a definição das estruturas, os *includes* necessários e o cabeçalho/assinatura das funções. Você não deverá alterar esse arquivo.
- **turma.c que conterá a implementação das funções solicitadas (e funções adicionais, caso julgue necessário). Este arquivo já contém o esqueleto geral das funções e alguns códigos implementados.**
- testaTurma.c que contém alguns testes executados sobre as funções implementadas.

Você deverá submeter **apenas** o arquivo turma.c, porém renomeie este arquivo para seuNúmeroUSP.c (por exemplo, 12345678.c) antes de submeter.

Não altere a assinatura de nenhuma das funções e não altere as funções originalmente implementadas (*inicializaTurma*, *reinicializaTurma*, *exibirTurma*, *print123*).

Preencha suas informações pessoas (nome e número USP) no cabeçalho do arquivo turma.c

Nenhuma das funções que você implementar deverá imprimir algo. Para *debugar* o programa você pode imprimir coisas, porém, na versão a ser entregue, suas funções não deverão imprimir nada (exceto pela função *exibirTurma* que já imprime algumas informações).

Você poderá criar novas funções (auxiliares), mas não deve alterar o arquivo turma.h. Adicionalmente, saiba que seu código será testado com uma versão diferente do arquivo testaTurma.c. Suas funções serão testadas individualmente e em conjunto.

Todos os trabalhos passarão por um processo de verificação de plágios. **Em caso de plágio, todos os alunos envolvidos receberão nota zero.**