ACH2023 – turma 03 – EP2 – 1º Semestre de 2022

Trabalho individual – entrega pelo sistema eDisciplinas até 04/12/2022

Gerenciamento de Turmas de Alunos (versão 2)

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

A estrutura será composta por **um arranjo de listas duplamente ligadas ordenadas circulares e com nó-cabeça**. Estas listas serão listas de **registros dos alunos(as)** e cada aluno(a) estará em uma lista diferente (todas ordenadas pelo número USP dos alunos da respectiva lista). Uma das listas conterá os alunos aprovados na disciplina, outra aqueles em recuperação e outra aqueles reprovados na disciplina.

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):

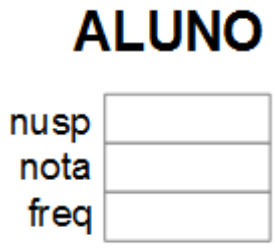
- criação/inicialização de uma turma;
- limpeza/reinicialização de uma turma;
- listagem de todos os alunos da turma;
- **consulta à quantidade de alunos na turma (tamanho);**
- **busca por um aluno na turma;**
- **inserção de um aluno na turma;**
- **exclusão de um aluno na turma;**
- **inserção da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

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

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

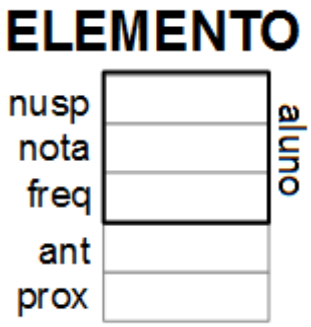
As informações dos alunos serão armazenadas na estrutura *ALUNO*, que contém três campos: *nusp* (número USP do aluno), *nota* (nota do aluno na turma) e *freq* (frequência do aluno na turma).

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



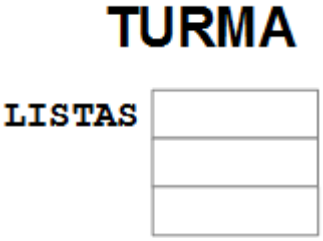
A estrutura básica das listas ligadas será o *ELEMENTO*, que contém três campos: *aluno* (campo do tipo ALUNO), *ant* e *prox* (ponteiros para, respectivamente, o endereço do elemento anterior e posterior ao atual).

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

typedef ELEMENTO* PONT;
```



A estrutura *TURMA* possui um arranjo de três posições: *LISTAS*, cada posição contém o ponteiro/referência para o nó-cabeça da lista ligada circular de elementos ordenada pelo número USP dos respectivos alunos. A lista ligada da posição zero (LISTAS[0]) conterá os elementos correspondentes aos alunos aprovados na disciplina (nota≥50 e frequência≥70). A lista ligada da posição um (LISTAS[1]) conterá os elementos correspondentes aos alunos que estão em recuperação na disciplina (nota≥30 e nota<50 e frequência≥70). Já a lista ligada da posição dois (LISTAS[2]) conterá os elementos correspondentes aos alunos reprovados na disciplina (frequência<70 ou [nota<30 antes da recuperação ou nota<50 depois da recuperação]). **Cada lista de elementos será duplamente ligada, ordenada, circular e possuirá um nó-cabeça.**

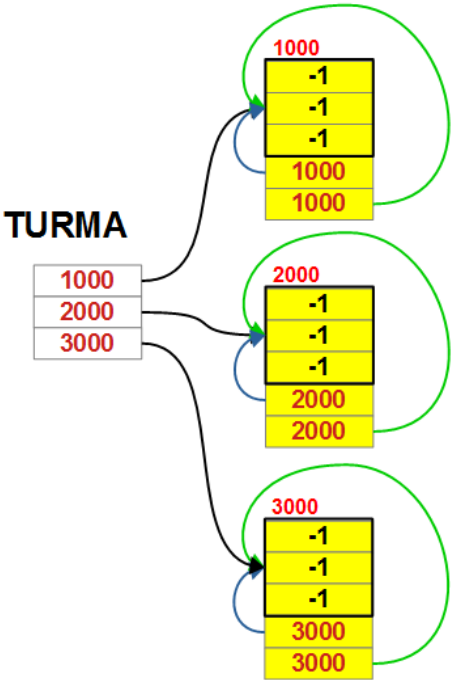
```
typedef struct {
     PONT LISTAS[3];
} TURMA;
```



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 nunca deverão ser apagados/excluídos.**

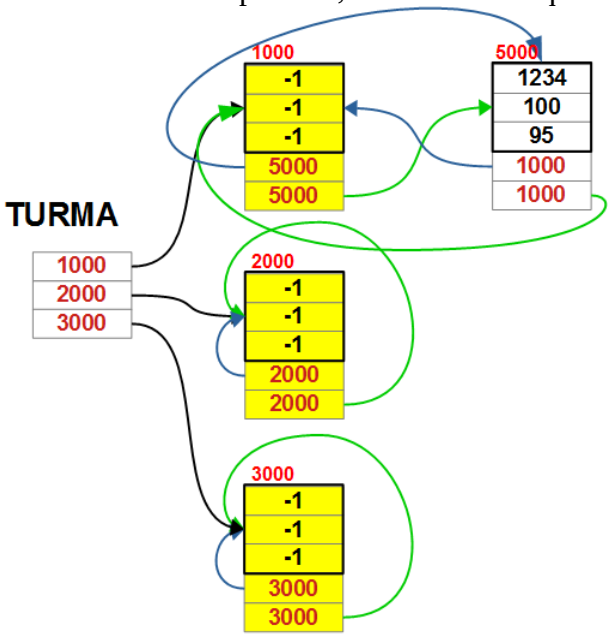
```
TURMA inicializaTurma(){
     TURMA t1;

     // criacao de um aluno ficticio que poder ser usado como sentinela
     ALUNO ficticio;
     ficticio.nusp = -1;
     ficticio.nota = -1;
     ficticio.freq = -1;

     // criacao dos tres nos-cabeca
     int c;
     for (c=0; c<3; c++){
          t1.LISTAS[c] = (PONT) malloc(sizeof(ELEMENTO));
          t1.LISTAS[c]->aluno = ficticio;
          t1.LISTAS[c]->prox = t1.LISTAS[c];
          t1.LISTAS[c]->ant = t1.LISTAS[c];
     }
     return t1;
}
```

Exemplo de turma recém-criada:

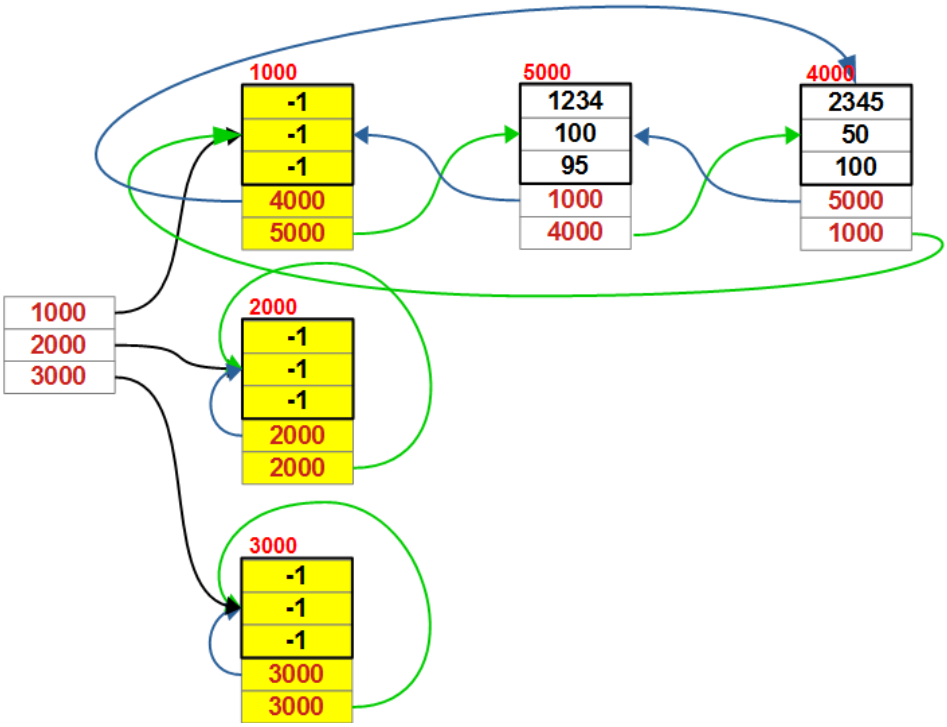


Ao se inserir o primeiro aluno na estrutura, este deverá ser incluído como próximo elemento do nó-cabeça, na lista correta, de acordo com sua nota e frequência.

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=50 e freq=100. Observe que as listas são ordenadas por número USP e este aluno também deverá ser inserido na primeira lista, pois também foi aprovado na disciplina.


1000
-1
-1
-1
4000
5000
5000
1234
100
95
1000
4000
4000
2345
50
100
5000
1000
1000
2000
3000
2000
-1
-1
-1
2000
2000
3000
-1
-1
-1
3000
3000

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 (deve somar os alunos das três listas).

PONT 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 ELEMENTO que contém o 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 (lembre-se que todas as listas são ordenadas de acordo com o número USP do aluno):
1a) alocação dinâmica de memória de uma estrutura do tipo ELEMENTO. Os campos do aluno do respectivo elemento deverão ser preenchidos com os valores recebidos como parâmetro pela função.
2a) este novo elemento deverá ser inserido na lista ligada correta, de acordo com a nota e a frequência do aluno (LISTAS[0] para alunos aprovados; LISTAS[1] para alunos em recuperação; e LISTAS[2] para alunos reprovados).
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 do elemento que aponta para o respectivo aluno na lista correta (não esqueça de acertar os ponteiros).
2a) liberação da memória do elemento da lista que apontava para o respectivo aluno.
3a) por fim, a função deverá retornar *true*.

bool inserirNotaRecuperacao(TURMA turma, int nusp, int nota)*: função que recebe o endereço de uma turma e o número USP e a nota de um aluno e deve: Retornar false caso este aluno não esteja presente na lista de alunos em recuperação (LISTAS[1]). Caso contrário, esse aluno deverá ter sua nota atualizada com o valor passado como parâmetro, ser retirado da lista de recuperação e inserido na lista correta, ordenada pelo número USP. A lista correta dependerá da nota do aluno: se maior ou igual que 50 deverá ir para a lista de aprovados (LISTAS[0]), caso contrário, para a lista de reprovados (LISTAS[2]). Após a remoção do aluno da lista de alunos em recuperação e inserção na lista correta (de acordo com o desempenho na recuperação), 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 04/12/2022 (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.**