**Instruções para a realização da avaliação prática**

1. *A prova terá duração de 150 minutos (2 horas e 30 minutos).*
2. *É responsabilidade do aluno salvar periodicamente o seu trabalho!!!!.*
3. *O nome do seu projeto e do seu programa-fonte devem ser iguais ao código da sua turma mais o número da sua matrícula e mais o seu primeiro nome. Exemplo: **33Z-1012983-maria**.*
4. *No início do seu programa-fonte, coloque, como comentários, o seu nome completo, sua matrícula, turm e o nome do seu professor.*
5. *Todos os arquivos mencionados no texto abaixo estão armazenados e/ou devem ser criados no diretório especificado pelo professor no início da prova.*
6. *Não é permitido destacar as folhas que compõem a prova (2 folhas).*
7. ***Antes de perguntar, leia atentamente o que está sendo pedido.***
8. *Quando terminar a prova, permaneça sentado e chame um professor ou um fiscal.*
9. *A nota da prova será entregue na data divulgada pelo seu professor.*
10. *O arquivo de saída não deverá ser entregue.*

**Questão única**

Uma instituição oferece cursos em 3 turnos: M→ Manhã, T→ Tarde, N→ Noite e os alunos inscritos em disciplinas EaD, comuns aos cursos, devem fazer a prova presencial, realizada em julho, no turno de seu curso. O comitê de organização da prova e alocação das salas de aulas, precisa saber quantos alunos realizarão as provas em cada um dos turnos.

O arquivo **disciplinas.txt** armazena a quantidade de alunos nas ( no máximo 15) diferentes disciplinas EaD, do seguinte modo:

- Disciplina (string de, no máximo, 10 caracteres válidos);
- Qt de alunos inscritos na disciplina em cada um dos 3 turnos (3 inteiros)

**Exemplo do arquivo inscritos.txt:**

| |
|---|
| **LPO001**<br>**400 200 1000** |

**Significado →** a disciplina LPO001 tem 400 alunos inscritos no turno manhã, 200 alunos inscritos no turno tarde e 1000 alunos inscritos no turno noite

No início do mês de junho, os alunos inscritos em alguma disciplina EaD podem solicitar a troca do turno para a realização da prova. O arquivo texto **solicitacoes.txt** armazena, em pares de linhas, a disciplina, a matrícula do aluno, o turno origem e o turno destino:

- Disciplina (string de, no máximo, 10 caracteres válidos);
- Matrícula do aluno ( inteiro) turno de origem ( M ou T ou N) turno destino( M ou T ou N)

**Exemplo do arquivo solicitacoes.txt**:

| |
|---|
| **LPO001**<br>**102030 M T** |

**Significado →**O aluno de matrícula 102030, matriculado no turno Manhã, deseja realizar a prova de LPO001 no turno Tarde.

Faça um programa MODULARIZADO que, utilizando as informações armazenadas nos 2 arquivos, gere o arquivo **ATUAL.TXT** com a quantidade de alunos que devem realizar as provas em cada um dos turnos em cada um das disciplinas.

→No final mostrar na tela a quantidade de salas em cada um dos turnos( considerando que as salas tem capacidade para 40 alunos

**Obrigatoriamente**, o seu programa deve utilizar as seguintes funções feitas por você:

a) função **busca()**: implementa o algoritmo de busca sequencial;
b) função **le_arq_disciplinas()**: realiza a leitura dos dados do arquivo **disciplinas.txt**, preenchendo os parâmetros recebidos;
c) função **atualiza_situacao**: uma de suas responsabilidades é fazer a leitura dos dados do arquivo **resultados.txt**, efetivando as operações realizadas;
d) função **gera_arq_atual()**: cria o arquivo **ATUAL.TXT** com as informações atualizadas;
e) função **descobre_pos_turnos()**: a função recebe o caracter que representa turno origem e o caracter que representa o turno destino e transforma-os em 0 se for manhã, , 1, se for tarde e 2 se for noite Dois dos parâmetros que esta função deverá receber são endereços de memória onde a função irá armazenar o número do torno origem e o número do turno destino.

| **disciplinas.txt** | **solicitacoes.txt** | **Atual.txt** |
|---|---|---|
| LPO001<br>100 200 300<br>LPO002<br>200 100 300<br>EMP001<br>1 2 3<br>EMP002<br>3 4 5 | LPO001<br>102030 M T<br>EMP001<br>102030 M T<br>LPO001<br>203010 T M<br>EMP002<br>203010 N M<br>LPO002<br>102323 N T<br>LPO001<br>103443 M N<br>LPO002<br>102124 T N<br>LPO001<br>102124 T N | LPO001 99 199 302<br>LPO002 200 100 300<br>EMP001 0 3 3<br>EMP002 4 4 4 |

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char detcol(char turno);
#define MAXCHAR 11
#define QTDturnos 3
#define MAXdisciplinas 15
int le_arq_disciplinas (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos]);
void atualiza_situacao (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos], int total);
int busca (char s[][MAXCHAR], int total, char chave[]);
void gera_arq_atual (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos], int total);
void descobre_pos_turnos (char orig, char dest, int *cOrig, int *cDest);
char detcol(char turno);

int main (void)
{
	char disciplinas[MAXdisciplinas][MAXCHAR];
	int mturnos[MAXdisciplinas][QTDturnos], total;
	total = le_arq_disciplinas (disciplinas, mturnos);
	atualiza_situacao(disciplinas, mturnos, total);
	gera_arq_atual(disciplinas, mturnos, total);
	return 0;
}

int le_arq_disciplinas (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos])
{
	FILE *in;
	int i, j;

	if ((in = fopen ("disciplinas.TXT", "rt")) == NULL)
	{
		puts ("Erro ao abrir o arquivo disciplinas.TXT");
		exit (1);
	}
	i = 0;
	while (fscanf (in, " %[^\n]", disciplinas[i]) == 1)
	{
		for (j = 0; j < QTDturnos; j++)
			fscanf (in, "%d", &mturnos[i][j]);
		i++;
	}
	fclose (in);
	return i;
}

void atualiza_situacao (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos], int total)
{
	FILE *in;
	int lin, colOrig,colDest,matr;
	char orig,dest, disc[MAXCHAR];

	if ((in = fopen ("solicitacoes.txt", "rt")) == NULL)
	{
		puts ("Erro ao abrir o arquivo solicitacoes.TXT");
		exit (1);
	}
	while (fscanf (in, " %[^\n]", disc) == 1)
	{
		fscanf (in, "%d %c %c", &matr, &orig, &dest);
		lin = busca (disciplinas, total, disc);
		if (lin != -1)
		{
			descobre_pos_turnos(orig,dest,&colOrig,&colDest);
			if(colOrig !=-1 && colDest != -1)
			{
				mturnos[lin][colOrig]--;
				mturnos[lin][colDest] ++;
			}
		}
```

```
	}
	fclose (in);
}

int busca (char s[][MAXCHAR], int total, char chave[])
{
	int pos;

	for(pos=0;pos < total;pos++)
	{
		if (strcmp(chave, s[pos]) == 0)
			return pos;
	}
	return -1;
}

void gera_arq_atual (char disciplinas[][MAXCHAR], int mturnos[][QTDturnos], int total)
{
	FILE *out;
	int i, j, maior, menor, soma;

	if ((out = fopen ("atual.TXT", "wt")) == NULL)
	{
		puts ("Erro ao abrir o arquivo FINAL.TXT");
		exit (1);
	}

	for (i = 0; i < total; i++)
	{
		fprintf (out, "\n%-10s", disciplinas[i]);
		for (j = 0; j < QTDturnos; j++)
		{
			fprintf (out, "%6d ", mturnos[i][j]);
		}
	}
	fclose (out);
}

char detcol(char turno)
{
	if(turno == 'm' || turno == 'M')
		return 0;
	if(turno == 't' || turno == 'T')
		return 1;
	if(turno == 'n' || turno == 'N')
		return 2;
	return -1;
}
void descobre_pos_turnos (char orig, char dest, int*cOrig, int*cDest)
{
	*cOrig= detcol(orig);
	*cDest=detcol(dest);
}
```