UNIFEI	Universidade Federal de Itajubá Instituto de Engenharia de Sistemas e Tecnologias da Informação-IESTI
8º Laboratório de ECOP12 – Estrutura de Dados Profª. Thatyana de Faria Piola Seraphim	

O objetivo do laboratório é implementar um simulador de escalonador de processos da CPU usando filas dinâmicas encadeadas.

A variável **filaDeProcessos** é definida como global para todas as funções desse programa, sendo que representa 3 filas de processos. Por convenção: `filaDeProcessos[0]` armazena processos com prioridade entre 0 e 3; `filaDeProcessos[1]` armazena processos com prioridade entre 4 e 6; filaDeProcessos[2] armazena processos com prioridade entre 7 e 9. Cada elemento nas filas representa um único processo na CPU do tipo noProcesso.

Abaixo são dadas algumas definições que serão usadas no programa.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>

#define NUM_FILAS 3
#define TAM_NOME 50

//estrutura de um processo
typedef struct no{
   float tamanho;
   char nome[TAM_NOME];
   int prioridade;
   struct no *prox;
}noProcesso;

//fila de processos a serem executados na CPU
noProcesso *filaDeProcessos[NUM_FILAS];

//vetor que indica a existencia de pelo menos um processo dessa prioridade
bool prioridadesProcessos[10];
```

a) Faça uma função chamada **inicializaFila** que não recebe parâmetros e não retorna nada. Essa função deve inicializar cada elemento do vetor chamado filaDeProcessos com o valor NULL.

b) Faça uma função chamada **inicializaVetorPrioridades** que não recebe parâmetros e não retorna nada. Essa função deve inicializar cada elemento do vetor chamado prioridadesProcessos com o valor *false*.

c) Faça uma função chamada **insereFilainsereFila** que não retorna nada e recebe como parâmetros: um tamanho do tipo float, que indica o tamanho do processo; o nome do tipo char, que indica o nome do processo; e um inteiro chamado prioridade, que indica a prioridade do processo de acordo com as informações indicadas acima. De acordo com a prioridade do processo, esse será inserido em uma das filas. Só lembrando, que o processo será sempre inserido no final da fila.

d) Faça uma função chamada **imprimeDadosListasDeProcessos** que não retorna nada e percorre todas as filas, imprimindo os dados de cada um dos processos, ou seja, número do processo, nome do processo, tamanho e prioridade desse processo. Esta função deve ainda contabilizar o número de processos e realizar a soma do tamanho dos processos de cada fila. Essas informações também devem ser impressas na tela.

e) Faça uma função chamada **removePrimeiroDaFila** que retorna verdadeiro caso o primeiro processo seja removido da fila, ou falso, caso contrário. Essa função deverá receber como parâmetro o índice de qual fila o processo será removido.

f) Faça uma função chamada **destroiFila** que não retorna nada e deverá liberar todo espaço de memória alocado na função de inserção.

g) Deverá ser realizada a leitura de um valor inteiro que indica o índice da fila que o processo deverá ser removido. O índice deverá ser passado para a função **removePrimeiroDaFila** para que o primeiro processo da fila informada seja removido. Em seguida, faça uma chamada à função que imprime os dados das filas para exibir a retirada com sucesso do primeiro processo da fila indicada. Faça a chamada para a função **destroiFila**. Inclua a definição do método *main* abaixo:

```
int main(int argc, char **argv){
   inicializaFila();
   inicializaVetorPrioridades();

   insereFila(1, "BrOfficeCalc", 3);
   insereFila(2.3, "CodeBlocks", 9);
   insereFila(0.5, "Calculadora", 1);
   insereFila(6.4, "Firefox", 7);
   insereFila(3.1, "PhpMyAdmin", 2);
   insereFila(2.9, "Kile", 4);
   insereFila(4.5, "Gimp", 9);
   insereFila(5.5, "BrOfficeWriter", 6);
   insereFila(5.8, "GCC", 2);

   imprimeDadosListasDeProcessos();

   return 0;
}//fim main
```