3ª Trabalho
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Entrega: 29/03/2021 via e-mail para jarbas_joaci@yahoo.com.br

Obs. 1ª - O trabalho é individual e não será recebido após a data mencionada. 2ª - Preferencialmente fazer o trabalho usando a IDE Dev-C++. Enviar todos os arquivos do projeto, exceto os executáveis (.exe).

1. Implemente a TAD "filaprio.h" com as funções apresentadas nos slides "Fila de Prioridades". Os protótipos das funções são: (5,0 pontos)

- `FilaP* filap_cria();` //cria uma fila de prioridade vazia;

- `int heap_maximum(FilaP* f);` //devolve o elemento de maior prioridade na fila;

- `int heap_extract_max(FilaP* f);` //devolve o elemento de maior prioridade e o retira da fila;

- `void heap_increase_key(FilaP* f, int ind, int chave);` //modifica a prioridade de determinado elemento no índice `ind` na fila (apenas se chave for maior);

- `void max_heap_insert(FilaP* f, int chave);` //insere um novo elemento na fila com prioridade `chave`.

Obs. `FilaP` é do tipo ponteiro para uma estrutura que contém dois campos: um vetor de inteiros `v` que guarda as prioridades e um inteiro `t_heap` que guarda o tamanho de heap. As prioridades são positivas.

A seguir, execute o seguinte programa

```
#include <stdio.h>
#include <stdlib.h>
#include "filaprio.h"

int main(void){

  FilaP* fp = filap_cria();

  int a;

  max_heap_insert(fp,44);
  max_heap_insert(fp,55);
  max_heap_insert(fp,21);
  max_heap_insert(fp,98);
  max_heap_insert(fp,92);
```

```
  max_heap_insert(fp,64);
  max_heap_insert(fp,42);

  printf(''Prioridade: %d\n'', heap_maximum(fp));
  a= heap_extract_max(fp);
  printf(''Prioridades: %d e %d\n'', a, heap_maximum(fp));
  a =heap_extract_max(fp);
  printf(''Prioridade: %d e %d\n'', a, heap_maximum(fp));
  heap_increase_key(fp,2,10);
  printf(''Elem. maior prioridade: %d\n'', heap_maximum(fp));
  heap_increase_key(fp,2,100);
  printf(''Elem. maior prioridade: %d\n'', heap_maximum(fp));

  system("PAUSE");

  return 0;

}
```

2. Implemente os algoritmos **BubbleSort**, **InsertionSort**, **QuickSort, MergeSort** e **HeapSort** e calcule o tempo médio de cada um para ordenar vetores com valores aleatórios de tamanho 10^n ($n \in \{1, 2, ..., 6\}$). Elabore um relatório com gráficos que relacionem o tamanho dos vetores com os tempos para ordená-los. (5,0 pontos)

Obs. O tempo deve ser dado em milissegundos.