Lista 2 – Estrutura de Dados 1 – 2023.1

Prof. Ana Luiza Bessa de Paula Barros
Ciência da Computação – UECE

1) Quais são as duas partes constituintes necessárias para a definição de um TAD?

2) Toda função que compõe um TAD deve receber necessariamente pelo menos um atributo. Qual é este atributo? Justifique sua resposta.

3) Crie os TADs indicados abaixo:

   a) Jogador de Futebol

   i) Cada jogador possui os campos: Nome, Jogos, Gols e Assistências

   ii) Implemente as operações: “Atribuir”, “Imprimir” e “BomJogador”

   b) Time de Futebol

   i) Os times possuem os atributos: Nome, Jogadores, Vitórias, Empates e Derrotas

   ii) Implemente as operações: “Atribuir”, “Imprimir” e “Pontuação” (um time de futebol recebe 3 pontos por vitória e 1 ponto por empate)

4) Crie um Tipo Abstrato de Dados para representar um **Número Complexo** $z = x + iy$, em que $i = -1$, sendo x a sua parte real e y a parte imaginária.

   Implemente funções para:

   a) Criar um número complexo
   b) Destruir um número complexo
   c) Realizar a soma de dois números complexos
   d) Realizar a multiplicação de dois números complexos

5) Crie um Tipo Abstrato de Dados para representar uma **Esfera**. Inclua as funções de inicialização necessárias e as operações que retornem o seu raio, a sua área e o seu volume.

6) Dado o TAD **Ponto** descrito abaixo:

```
** Libera a memória de um ponto previamente criado.
*/
void libera (Ponto* p);
/* Função acessa
** Devolve os valores das coordenadas de um ponto
*/
void acessa (Ponto* p, float* x, float* y);
/* Função atribui
** Atribui novos valores às coordenadas de um ponto
*/
void atribui (Ponto* p, float x, float y);
/* Função distancia
** Retorna a distância entre dois pontos
*/
float distancia (Ponto* p1, Ponto* p2);
```

```
#include <stdlib.h> /* malloc, free, exit */
#include <stdio.h> /* printf */
#include <math.h> /* sqrt */
#include "ponto.h"

struct ponto {
  float x;
  float y;
};

Ponto* cria (float x, float y) {
  Ponto* p = (Ponto*) malloc(sizeof(Ponto));
  if (p == NULL) {
    printf("Memória insuficiente!\n");
    exit(1);
  }
  p->x = x;
  p->y = y;
  return p;
}
void libera (Ponto* p) {
  free(p);
}

void acessa (Ponto* p, float* x, float* y) {
  *x = p->x;
  *y = p->y;
}
void atribui (Ponto* p, float x, float y) {
  p->x = x;
  p->y = y;
}

float distancia (Ponto* p1, Ponto* p2) {
  float dx = p2->x - p1->x;
  float dy = p2->y - p1->y;
  return sqrt(dx*dx + dy*dy);
}
```

a) Escreva um programa que faça uso do TAD **Ponto**

b) Acrescente novas operações ao TAD **Ponto**, tais como soma e subtração de pontos