Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Disciplina: Algoritmos 1
Professora: Emanoeli Madalosso
Curso de Engenharia de Computação

Nome: _____

Avaliação 1 - 25/04/2018

1) Escreva uma função que recebe dois números inteiros, “a” e “b”. A função deve trocar os valores: “a” deve ficar com o valor de “b” e “b” deve ficar com o valor de “a”. Forneça um exemplo de como a sua função deve ser chamada.

2) Explique para que servem as funções malloc() e calloc(), disponíveis na stdlib.h, citando a diferença entre as duas.

3) Um aluno de Algoritmos 1 está resolvendo um exercício onde precisa cadastrar e imprimir os dados de um livro. Ele deve usar o Tipo Abstrato de Dados (TAD) inicial fornecido no exercício:
Livro.h:

```
typedef struct livro Livro;
#define TAMSTR 50
```

Livro.c:

```
#include "Livro.h"

struct livro {
  char titulo[TAMSTR];
  char autor[TAMSTR];
  int edicao;
  int num_paginas;
};
```

Em seu arquivo usa_tad.c, o aluno fez o seguinte código:

```
#include <stdio.h>
#include <string.h>
#include "Livro.h"

void main(void) {
  Livro l;
  char titulo[TAMSTR];
  char autor[TAMSTR];
  int edicao;
  int num_paginas;
```

```c
    // lendo dos dados do livro
    printf("Informe o título do livro: ");
    scanf("%[^\n]s", titulo);
    getchar();
    printf("Informe o autor do livro: ");
    scanf("%[^\n]s", autor);
    getchar();
    printf("Informe a edição do livro: ");
    scanf("%d", &edicao);
    printf("Informe o número de páginas do livro: ");
    scanf("%d", &num_paginas);

    // cadastrando o livro
    strcpy(l.titulo, titulo);
    strcpy(l.autor, autor);
    l.edicao = edicao;
    l.num_paginas = num_paginas;

    // imprimindo os dados do livro cadastrado
    printf("Título: %s\n", l.titulo);
    printf("Autor: %s\n", l.autor);
    printf("Edição: %d\n", l.edicao);
    printf("Páginas: %d\n", l.num_paginas);
}
```

O código feito pelo aluno será capaz de cadastrar um livro e imprimir seus dados corretamente? Se não, aponte os problemas do código e explique como ele poderia ser corrigido (respeitando os conceitos de TADs).

4) Responda as perguntas:
a) Diferencie lista, pilha e fila.
b) Sabendo as características de cada uma, qual delas você usaria para resolver o problema abaixo? Forneça uma explicação ou pseudocódigo de como você resolveria.
Problema do maior item mais próximo:

Para cada item de uma sequência, o maior item mais próximo (do lado direito) deve ser encontrado. Se não houver um item maior, considera-se -1. Exemplos:

Sequência de entrada:		Sequência de entrada:	
4, 5, 2, 25		13, 7, 6, 12	
Saída:		Saída:	
4	5	13	-1
5	25	7	12
2	25	6	12
25	-1	12	-1

5) Dois alunos de Algoritmos 1 estavam fazendo um trabalho sobre filas implementadas por meio de vetores. Os dois alunos têm arquivos Fila.h iguais:
Arquivo Fila.h:

```
typedef struct item Item;
typedef struct fila Fila;

#define MAXTAM 5

Fila * cria_fila_vazia();
void enfileira(Fila* f, int chave);
void desenfileira(Fila* f);
void libera(Fila *f);
```

Os dois alunos alunos usaram estruturas iguais e as funções cria_fila_vazia() e libera() também são iguais:
Arquivo Fila.c:

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

struct item {
  int chave;
  // demais campos
};
struct fila {
  Item item[MAXTAM];
  int primeiro;
  int ultimo;
  int tamanho;
};
```

```
Fila * cria_fila_vazia() {
  Fila *f = malloc(sizeof(Fila));
  f->primeiro = 0;
  f->ultimo = 0;
  f->tamanho = 0;
  return f;
}

void libera(Fila *f) {
  free(f);
```

O aluno A implementou as funções de enfileira() e desenfileira() da seguinte forma:

```
void enfileira(Fila* f, int chave) {
  if(f->ultimo == MAXTAM) {
      printf("Fila cheia!\n");
      return;
  }
  Item novo_item;
  novo_item.chave = chave;
  f->item[f->ultimo] = novo_item;
  f->ultimo++;
  f->tamanho++;
  printf("Item %d enfileirado com sucesso!\n", chave);
}
```

```c
void desenfileira(Fila* f) {
  if(f->tamanho == 0) {
      printf("Fila vazia!\n");
      return;
  }
  f->primeiro++;
  f->tamanho--;
  printf("Item desenfileirado com sucesso!\n");
}
```

O aluno B implementou as funções de enfileira() e desenfileira() da seguinte forma:

```c
void enfileira(Fila* f, int chave) {
  if(f->tamanho == MAXTAM) {
      printf("Fila cheia!\n");
      return;
  }
  Item novo_item;
  novo_item.chave = chave;
  f->item[f->ultimo] = novo_item;
  f->ultimo = (f->ultimo + 1) % MAXTAM;
  f->tamanho++;
  printf("Item %d enfileirado com sucesso!\n", chave);
}

void desenfileira(Fila* f) {
  if(f->tamanho == 0) {
      printf("Fila vazia!\n");
      return;
  }
  f->primeiro = (f->primeiro + 1) % MAXTAM;
  f->tamanho--;
  printf("Item desenfileirado com sucesso!\n");
}
```

Verifique a saída de cada aluno ao aplicar as seguintes operações:

enfileira(f, 2);
enfileira(f, 10);
desenfileira(f);
enfileira(f, 23);
enfileira(f, 5);
desenfileira(f);
enfileira(f, 11);
enfileira(f, 34);
enfileira(f, 50);

Saída do aluno A:

```
Item 2 enfileirado com sucesso!
Item 10 enfileirado com sucesso!
Item desenfileirado com sucesso!
Item 23 enfileirado com sucesso!
Item 5 enfileirado com sucesso!
Item desenfileirado com sucesso!
Item 11 enfileirado com sucesso!
Fila cheia!
Fila cheia!
```

Saída do aluno B:

```
Item 2 enfileirado com sucesso!
Item 10 enfileirado com sucesso!
Item desenfileirado com sucesso!
Item 23 enfileirado com sucesso!
Item 5 enfileirado com sucesso!
Item desenfileirado com sucesso!
Item 11 enfileirado com sucesso!
Item 34 enfileirado com sucesso!
Item 50 enfileirado com sucesso!
```

Responda as perguntas:

- Quantos itens cada aluno conseguiu guardar na fila?
- Qual a principal diferença entre as implementações do aluno A e do aluno B?
- Qual implementação você considera mais vantajosa? Por quê?

6) Você está prestes a fazer uma viagem de avião e está preocupado com o peso da sua bagagem. Você faz uma lista com todos os itens que deseja levar, com seus respectivos pesos:

Item	**Peso (Kg)**
Carregador do celular	0,100
Meia	0,090
Camiseta	0,300
Calça	0,400
Jaqueta	0,600
Escova	0,225
Lembrança para a avó	1,000
Notebook	2,000
Livro	0,400

Contando com o peso da mala (2,000 Kg), o peso total da bagagem será 7,115 Kg. O limite para a sua bagagem é de apenas 5 Kg. Você resolve então ir removendo os itens mais pesados até conseguir ficar dentro do limite de peso:

1. O item mais pesado é o notebook. Ao removê-lo, sua bagagem fica com 5,125 Kg, ainda acima do limite.
2. O item mais pesado agora é a lembrança para a sua avó. Você desiste de levar a lembrança e fica com 4,125 Kg, ficando dentro do limite.

Seguindo esta ideia, escreva um programa que tenha uma entrada onde a primeira linha contém um valor P (limite de peso da bagagem), a segunda linha contém um valor M

(peso da mala), a terceira linha contém um valor N (número de itens na mala) e as demais linhas contém os códigos e pesos dos N itens. Exemplo de entrada:

```
4.000
1.500
8
1 2.000
2 0.500
3 0.250
4 1.000
5 1.500
6 0.750
7 0.950
8 0.800
```

A saída deve mostrar quais itens foram removidos, respeitando a ordem de remoção. Exemplo de saída:

```
1 2.000 5 1.500 4 1.000 7 0.950
```

Você pode usar o tipo lista fornecido abaixo como base para resolver esse problema. Crie as funções que achar necessário.

Lista.h:

```c
#define MAXTAM 100
typedef struct item Item;
typedef struct lista Lista;

Lista * cria_lista_vazia();
int verifica_lista_vazia(Lista *l);
int verifica_lista_cheia(Lista *l);
void adiciona_item_fim_lista(Lista *l, int codigo, float peso);
void imprime_lista(Lista *l);
void libera_lista(Lista *l);
```

Lista.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "Lista.h"

struct item {
    int codigo;
    float peso;
};
struct lista {
    Item item[MAXTAM];
    int ultimo;
};
```

```c
Lista * cria_lista_vazia() {
    Lista *l = malloc(sizeof(Lista));
    l->ultimo = -1;
    return l;
}

// retorna 1 se a lista está vazia ou 0 se não está vazia
int verifica_lista_vazia(Lista *l) {
    return l->ultimo == -1;
}

// retorna 1 se a lista está cheia ou 0 se não está cheia
int verifica_lista_cheia(Lista *l) {
    return l->ultimo == MAXTAM - 1;
}

void adiciona_item_fim_lista(Lista *l, int codigo, float peso) {
    if(verifica_lista_cheia(l)){
        printf("Lista cheia!\n");
        return;
    }
    Item novo_item;
    novo_item.codigo = codigo;
    novo_item.peso = peso;
    l->ultimo++;
    l->item[l->ultimo] = novo_item;
}

void imprime_lista(Lista *l) {
    int tam = l->ultimo + 1;
    int i;
    for(i = 0; i < tam; i++)
        printf("%d %.3f ", l->item[i].codigo, l->item[i].peso);
}

void libera_lista(Lista *l) {
    free(l);
}
```