AP1 ESTRUTURAS DE DADOS

Wendell Lucas Souza Batista - 516107.

Q7.

A)

```
INT QTD_MENORES_LISTA(LISTA *l, INT x){
    LISTA *LAUX = l;
    INT CONT = 0;
    WHILE(LAUX != NULL){
        IF(LAUX->INFO < x){
            CONT++;
        }
        LAUX = LAUX->PROX;
    }
    RETURN CONT;
}
```

2,0

B)

```
LISTA* INSERE_FIM_LISTA(LISTA *l, INT x){
    LISTA *LAUX = l;
    LISTA *LN = (LISTA*) MALLOC(SIZEOF(LISTA));
    LN->INFO = X;
    LN->PROX = NULL;
    IF(l == NULL){
        RETURN LN;
    }
    WHILE(LAUX != NULL){
        LAUX = LAUX->PROX;
    }
    LAUX->PROX = LN;
    RETURN l;
};
```

laux → prox

1,5

2.

A)

```
VOID PILHA_PUSH_MAIOR (PILHA *P, INT INFO) {
    LISTA *LAUX = P->PRIM;
    IF(LAUX->INFO < INFO || P->PRIM == NULL) {
     LISTA *LN = (LISTA*) MALLOC (SIZEOF(LISTA));
     LN->INFO = INFO
     LN->PROX = P->PRIM;
     P->PRIM = LN;
    }
};
```

→ Se laux == null?

4,8

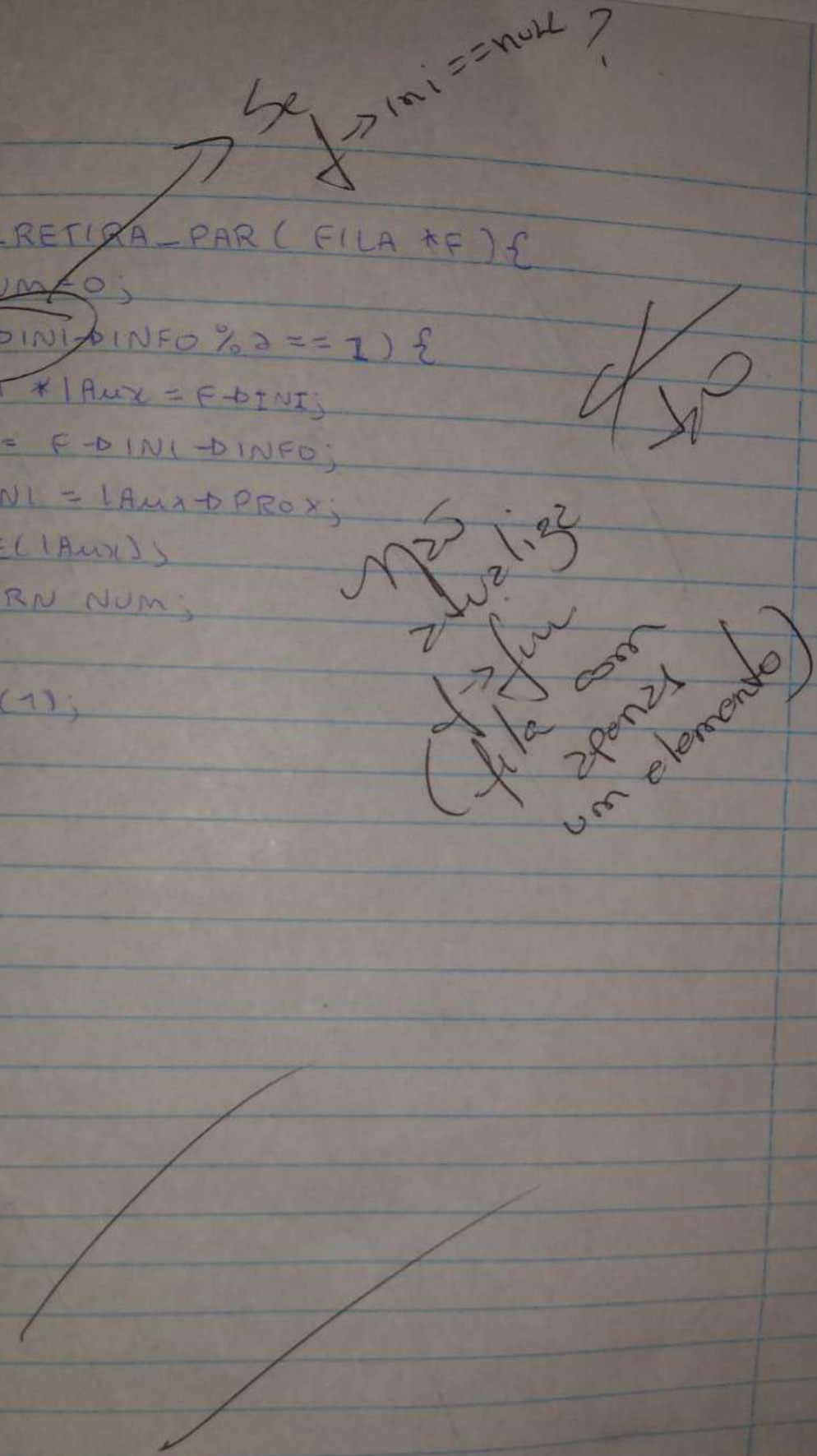
B)

```
INT PILHA_SOMA(PILHA *P) {
    LISTA *LAUX = P->PRIM;
    INT SOMA=0;
    WHILE(LAUX != NULL) {
      SOMA = SOMA + LAUX->INFO;
      LAUX = LAUX->PROX;
    }
    RETURN SOMA;
}.
```

✓ 2,0

03.

```
INT FILA_RETIRA_PAR ( FILA *F ){
   INT NUM=0;
   IF( F->INI->INFO % 2 == 1 ) {
    LISTA *|Aux = F->INI;
    NUM = F->INI->INFO;
    F->INI = |Aux->PROX;
    FREE(|Aux);
    RETURN NUM;
   }
   F)(i0(1);
}
```



1ª Avaliação Parcial
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Nome: Wendell Luan Sousa Batista Data 09/05/2023

1ª) Considere que um nó de uma lista encadeada é dado por:

```
typedef struct lista Lista;
struct lista {
  int info;
  Lista *prox;
};
```

a) Implemente uma função que tenha como valor de retorno a quantidade de nós com valores **menores** que *x* em uma lista encadeada. O protótipo da função deve ser: (2,0 pontos)

```
int qtd_menores_lista(Lista* l, int x);
```

b) Implemente uma função que insira um valor num **novo nó** no **fim** de uma lista. O protótipo da função deve ser: (2,0 pontos)

```
Lista* insere_fim_lista(Lista* l, int x);
```

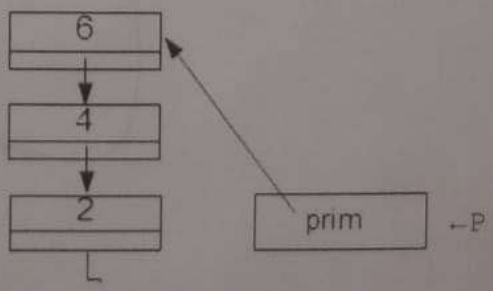
2ª) Considere uma pilha implementada por meio de lista encadeada, conforme as descrições de estruturas abaixo:

```
typedef struct lista Lista;

typedef struct pilha Pilha;

struct lista{
   int info;
   Lista *prox;
};

struct pilha{
   Lista *prim;
};
```



Exemplo de Pilha

a) Escreva uma função que acrescente um elemento ao topo da pilha apenas se ele for **maior** que o topo já existente. O protótipo da função deve ser: (2,0 pontos)

```
void pilha_push_maior(Pilha *p, int info);
```

Obs. se a pilha estiver inicialmente vazia, qualquer valor será aceito como topo da pilha.

b) Escreva uma função que retorne a **soma** dos valores em uma pilha. O protótipo da função deve ser: (2,0 pontos)

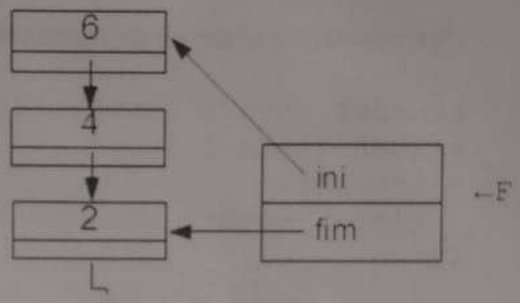
```
int pilha_soma(Pilha *p);
```

3ª) Considere uma fila implementada por meio de lista encadeada, conforme as descrições de estruturas abaixo:

```
typedef struct lista Lista;
typedef struct fila Fila;

struct lista{
   int info;
   Lista *prox;
};

struct fila{
   Lista *ini;
   Lista *fim;
};
```



Exemplo de Fila

Escreva uma função que retire um elemento do início da fila e que retorne o seu valor **apenas se ele for ímpar (caso contrário, a fila permanece inalterada)**. O protótipo da função deve ser: (2,0 pontos)

```
int fila_retira_impar(Fila *f);
```