INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE ED e LAB. ED
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Avaliativa de ED e Lab. ED - Unidade II

Instruções

LISTAS DIVERDIDA MENTE

Responda às questões abaixo.

O aluno poderá fazer uso de qualquer material visto em sala de aula.

Atenção: Escreva suas respostas em cor de raiva (vermelho).

Questões

A Sala de Controle estava em polvorosa. As emoções de Riley disputavam a todo instante para decidir quem estaria no controle. Do lado de fora, não havia quem a pudesse suportar: ora dominada pela **alegria**, ora em furor de **raiva**; se o dia não fosse como esperado, batia a **tristeza**; e se achasse que não estava bem preparada para um prova, logo vinha o **medo**. E ainda tinha aqueles garotos medíocres da escola, que só falavam bobagens, que **nojo**.

Ao cabo de um dia agitado, Riley decide trancafiar-se em seu quarto, a fim de listar e compreender o seu comportamento recente. É preciso fazer alguma coisa.

Riley precisa de sua ajuda, antes que novas emoções entrem em cena para deixá-la ainda mais descontrolada.

Questão 1) A primeira tarefa de Riley é organizar uma Lista Linear para inserir suas emoções na ordem em que ela for se lembrando. Sabendo-se que, no mundo de Riley, as emoções não passam de 10, que implementação de Lista Linear seria mais adequada para o problema em questão? Por quê? (1 ponto)

R: Ela teria que implementar uma Lista Linear não ordenada ou simples é adequada para a situação, pois a lista permite inserções e remoções naturalmente, enquanto não possui nenhum tipo de ordem definida, dessa forma, consegue resolver o problema de Riley..

Questão 2) Refletindo sobre o último mês, Riley fez uma Lista Sequencial contendo os dias do mês de Julho em que a **tristeza** foi a emoção dominante sobre ela. Tentou fazer de cabeça, mas depois lembrou-se que se divertiu com suas amigas em todos os domingos do mês. Escreva uma função chamada domingos_divertidos, que recebe uma Lista Sequencial e remove todos os domingos de Julho da lista. (2 pontos)

```c
#include <stdio.h>

#include <stdlib.h>



#define MAX 31



typedef struct {

    int *dias;

    int posicao;
```

```c
} MES;


void inicializar(MES *julho) {
    julho->dias = malloc(sizeof(int) * MAX);
    julho->posicao = 0;
}


void inserir(MES *julho, int elemento) {
    if (julho->posicao < MAX) {
        julho->dias[julho->posicao++] = elemento;
    } else {
        printf("tá cheia\n");
    }
}


int buscar(MES *julho, int elemento) {
    for (int i = 0; i < julho->posicao; i++) {
        if (julho->dias[i] == elemento) {
            return i;
        }
    }
    return -1;
}


void remover(MES *julho, int elemento) {
    int p = buscar(julho, elemento);
```

```
    if (p == -1) return;
    for (int i = p; i < julho->posicao - 1; i++) {
        julho->dias[i] = julho->dias[i + 1];
    }
    julho->posicao--;
}



void domingo_divertidos(MES *mes) {
    int domingos[] = {7, 14, 21, 28};
    int num_domingos = sizeof(domingos) / sizeof(domingos[0]);

    for (int i = 0; i < mes->posicao; ) {
        int dia_atual = mes->dias[i];
        int is_domingo = 0;
        for (int j = 0; j < num_domingos; j++) {
            if (dia_atual == domingos[j]) {
                is_domingo = 1;
                break;
            }
        }
        if (is_domingo) {
            remover(mes, dia_atual);
        } else {
            i++;
        }
```

```
    }
}

void imprimir(MES *julho) {
    for (int i = 0; i < julho->posicao; i++) {
        printf("%d ", julho->dias[i]);
    }
    printf("\n");
}

int main() {
    MES julho;
    inicializar(&julho);

    inserir(&julho, 1);
    inserir(&julho, 7);
    inserir(&julho, 8);
    inserir(&julho, 14);
    inserir(&julho, 15);
    inserir(&julho, 21);
    inserir(&julho, 22);
    inserir(&julho, 28);
    inserir(&julho, 29);

    printf("Lista antes das lembrancas:\n");
    imprimir(&julho);
```

```
    domingo_divertidos(&julho);


    printf("Lista depois das lembrancas:\n");
    imprimir(&julho);


    return 0;
}
```

Questão 3) Em determinado momento, ainda reflexiva, Riley percebeu que quando duas emoções se enfrentavam uma sempre saia vencedora: a emoção cujo comprimento de caracteres é o menor. Isso posto, escreva uma função chamada embate_emocoes que recebe duas Listas Simplesmente Encadeadas (de mesmo tamanho) e retorna uma terceira Lista, contendo as emoções vencedoras (comparando-se as emoções na mesma ordem de cada lista: a primeira da lista 1 com a primeira da lista 2; a segunda da lista 1 com a segunda da lista 2 etc.). Considere o registro nó como o seguinte: (2 pontos)

```
struct NO {
   char emocao[18];
   struct NO *prox;
};
```

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


typedef struct sNO {

    char emocao[18];

    struct sNO *prox;

} NO;
```

```c
NO* inicializar(char* emocao) {
    NO* init = (NO*) malloc(sizeof(NO));
    strncpy(init->emocao, emocao, 18 - 1);
    init->emocao[18 - 1] = '\0';
    init->prox = NULL;
    return init;
}

void inserir(NO **inicio, char* emocao) {
    NO *novo = inicializar(emocao);
    if (*inicio == NULL) {
        *inicio = novo;
    } else {
        NO *aux = *inicio;
        while (aux->prox != NULL) {
            aux = aux->prox;
        }
        aux->prox = novo;
    }
}

void imprimir(NO *inicio) {
    NO *aux = inicio;
    printf("[ ");
    while (aux != NULL) {
```

```
        printf("%s ", aux->emocao);
        aux = aux->prox;
    }
    printf("]\n");
}

char* emocao_vencedora(char *emocao_lista1, char *emocao_lista2) {
          return  (strlen(emocao_lista1)  <=  strlen(emocao_lista2))  ?
emocao_lista1 : emocao_lista2;
}

NO* embate_emocoes(NO *lista1, NO *lista2) {
    NO *resultado = NULL;
    NO **ultimo = &resultado;

    while (lista1 != NULL && lista2 != NULL) {
                  char  *vencedora  =  emocao_vencedora(lista1->emocao,
lista2->emocao);
        inserir(ultimo, vencedora);
        ultimo = &((*ultimo)->prox);
        lista1 = lista1->prox;
        lista2 = lista2->prox;
    }

    return resultado;
}
```

```c
int main() {
    NO *lista1 = NULL;
    NO *lista2 = NULL;

    inserir(&lista1, "alegria");
    inserir(&lista1, "tristeza");
    inserir(&lista1, "raiva");
    inserir(&lista1, "medo");

    inserir(&lista2, "feliz");
    inserir(&lista2, "deprimido");
    inserir(&lista2, "furia");
    inserir(&lista2, "ansiedade");

    printf("Lista 1:\n");
    imprimir(lista1);

    printf("Lista 2:\n");
    imprimir(lista2);

    NO *lista_vencedora = embate_emocoes(lista1, lista2);

    printf("Lista Vencedora:\n");
    imprimir(lista_vencedora);

    return 0;
```

```
}
```

Questão 4) Finalmente, após as coisas ficarem mais claras e organizadas em sua cabeça, Riley decidiu organizar suas emoções em uma Lista Duplamente Encadeada, mantendo uma emoção junto do número de vezes em que ela assumiu o controle. Considerando os registros a seguir como base:

```
struct EMOCAO {
   char nome[18];
   int freq;
};

struct NO {
   struct EMOCAO emocao;
   struct NO *ant, *prox;
};

//[...]
```

a) Escreva uma função chamada emocao_no_controle, que recebe uma Lista Duplamente Encadeada e uma string contendo a emoção. Essa função deverá incrementar em uma unidade o campo freq da referida emoção. Assuma que a lista não conterá emoções repetidas. (2 pontos)

b) Escreva uma função chamada emocao_dominante que retorna um ponteiro para o nó da emoção mais frequente. (1,5 ponto)

c) Escreva uma função chamada emocao_submissiva que retorna um ponteiro para o nó da emoção menos frequente. (1,5 ponto)

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



typedef struct sEMOCAO {

    char nome[18];

    int freq;

} EMOCAO;



typedef struct sNO {
```

```c
    EMOCAO emocao;
    struct sNO *ant, *prox;
} NO;


NO* criar_no(const char* nome) {
    NO* novo = (NO*) malloc(sizeof(NO));

    strncpy(novo->emocao.nome, nome, 18 - 1);
    novo->emocao.nome[18 - 1] = '\0';
    novo->emocao.freq = 1;
    novo->ant = NULL;
    novo->prox = NULL;
    return novo;
}

void emocao_no_controle(NO *inicio, const char *nome) {
    NO *atual = inicio;
    while (atual) {
        if (strcmp(atual->emocao.nome, nome) == 0) {
            atual->emocao.freq++;
            return;
        }
        atual = atual->prox;
    }
}
```

```c
NO* emocao_dominante(NO *inicio) {
    NO *atual = inicio;
    NO *dominante = NULL;
    int max_freq = -1;

    while (atual) {
        if (atual->emocao.freq > max_freq) {
            max_freq = atual->emocao.freq;
            dominante = atual;
        }
        atual = atual->prox;
    }

    return dominante;
}

NO* emocao_submissiva(NO *inicio) {
    NO *atual = inicio;
    NO *submissiva = NULL;
    int min_freq = INT_MAX;

    while (atual) {
        if (atual->emocao.freq < min_freq) {
            min_freq = atual->emocao.freq;
            submissiva = atual;
        }
```

```
        atual = atual->prox;
    }

    return submissiva;
}

void inserir(NO **inicio, NO **fim, const char *nome) {
    NO *novo = criar_no(nome);
    if (*inicio == NULL) {
        *inicio = *fim = novo;
    } else {
        (*fim)->prox = novo;
        novo->ant = *fim;
        *fim = novo;
    }
}

void imprimir(NO *inicio) {
    NO *aux = inicio;
    printf("[ ");
    while (aux) {
          printf("%s Tomou controle - %d vezes | ", aux->emocao.nome,
aux->emocao.freq);
        aux = aux->prox;
    }
    printf("]\n");
}
```

```c
int main() {
    NO *inicio = NULL;
    NO *fim = NULL;

    inserir(&inicio, &fim, "Alegria");
    inserir(&inicio, &fim, "Medo");
    inserir(&inicio, &fim, "Raiva");
    inserir(&inicio, &fim, "Tristeza");

    emocao_no_controle(inicio, "Raiva");
    emocao_no_controle(inicio, "Raiva");
    emocao_no_controle(inicio, "Alegria");

    printf("Emocoes:\n");
    imprimir(inicio);

    NO *dominante = emocao_dominante(inicio);
    if (dominante) {
            printf("A emocao Dominante e %s com frequencia %d\n",
dominante->emocao.nome, dominante->emocao.freq);
    }

    NO *submissiva = emocao_submissiva(inicio);
    if (submissiva) {
            printf("A emocao Submissiva e %s com frequencia %d\n",
submissiva->emocao.nome, submissiva->emocao.freq);
```

```
    }


    return 0;
}
```

“Um bom plano tem várias partes.”

- Ansiedade