Nome: Matheus Goulart Ranzani
RA: 800278

Simulação da Prova 1

Questão 1

Boolean Vazia(variável por referência F do tipo Fila)
// deve retornar verdadeiro se a fila não tiver nenhum elemento; falso caso contrário.

```
int vazia(Fila *F) {
    return F->primeiro == NULL && F->ultimo == NULL;
}
```

Retira(variável por referência F do tipo Fila, variável por referência X do tipo Elemento, variável por referência Erro tipo boolean)
// retira 1 elemento da fila F. Erro deve retornar verdadeiro se a fila não tiver nenhum elemento para ser retirado; falso caso contrário.

```
void retira(Fila *F, int *X, int *Erro) {
    if (vazia(F)) {
        *Erro = 1;
    } else {
        *Erro = 0;
        *X = F->primeiro->info;

        NodePtr primeiro = F->primeiro;
        NodePtr proximo = F->primeiro->dir;

        // Se a fila tiver apenas um elemento
        if (F->primeiro == F->ultimo) {
            F->primeiro = NULL;
            F->ultimo = NULL;

            return;
        }

        // Se a fila tiver apenas dois elementos
        if (proximo->dir == primeiro) {
            proximo->dir = proximo;
        }

        proximo->esq = F->ultimo;
        F->ultimo->dir = proximo;
        F->primeiro = proximo;

        free(primeiro);
    }
}
```

Destroi(variável por referência F do tipo Fila)
// desaloca (remove) todos os elementos da fila.

```
void destroi(Fila *F) {
    if (vazia(F)) {
        return;
    }

    NodePtr p_aux = F->primeiro;

    do {
        free(p_aux);
        p_aux = p_aux->dir;
    } while (p_aux != F->primeiro);

    // Se eu coloco isso aparece o erro "double free or corruption (out)"
    // free(F);

    // Se eu coloco esses "free()" não aparece o erro, mas precisa deles?
    free(F->primeiro);
    free(F->ultimo);
}
```

Questão 2

Insere (variável por referência **FV** do tipo FilaDeVacinacao; variável **Idade** do tipo inteiro);

/* insere 1 pessoa com a idade fornecida como parâmetro na fila FV. Desconsidere a possibilidade de a estrutura estar cheia. Tipo FilaDeVacinacao = um ponteiro do tipo NodePtr (ponteiro para nó). Deve-se inserir as pessoas mais velhas antes das mais jovens. Pessoas que tenham a mesma idade em anos, devem ser inseridas após as demais que já estão na fila */

```
void insere(FilaDeVacinacao *FV, int Idade) {
    NodePtr p = malloc(sizeof(NodePtr));
    p→info = Idade;

    NodePtr p_aux = FV→ptr;

    // Fila vazia
    if (FV→ptr == NULL) {
        p→next = p;
        FV→ptr = p;
    } else if (p→info ≥ FV→ptr→info) {
        // Idade maior ou igual que a do começo da fila
        while (p_aux→next ≠ FV→ptr) {
            p_aux = p_aux→next;
        }

        // Passa a ser o começo da fila
        p→next = FV→ptr;
        p_aux→next = p;
        FV→ptr = p;
    } else {
        // Idade no meio ou no fim da fila
        while (p_aux→next ≠ FV→ptr && p→info < p_aux→next→info) {
            p_aux = p_aux→next;
        }

        p→next = p_aux→next;
        p_aux→next = p;
    }
}
```

Questão 3

Int ItensCompradosErroneamente(ListaCadastral Carrinho de Compras, ListaCadastral ListaDeComprasOriginal);
// Calcula e retorna o número de itens comprados erroneamente, ou seja, itens que constam do carrinho de compras e não constam da lista original.

```
int itens_comprados_erroneamente(
    ListaCadastral carrinho_de_compras, ListaCadastral lista_de_compras_original
) {
    elemento item_carrinho;
    int tem_elemento;
    int resultado = 0;

    pega_o_primeiro(carrinho_de_compras, item_carrinho, tem_elemento);

    while (tem_elemento) {
        if (!esta_na_lista(lista_de_compras_original, item_carrinho)) {
            resultado++;
        }

        pega_o_proximo(carrinho_de_compras, item_carrinho, tem_elemento);
    }

    return resultado;
}
```