Nome: Matheus Goulart Ranzani
RA: 800278

Simulação da Prova 2

Questão 1

void Remove (variável por referência R do tipo ABB, Variável X do tipo int, variável por referência Ok do tipo bool);
/* esta função deve procurar X na ABB R e, caso encontrar, deve remover da árvore e retornar Ok = true. Caso não encontrar, Ok deve retornar false. O tipo ABB é análogo ao tipo NodePtr, ou seja, ponteiro para o nó da árvore */

```
void Remove(ABB* R, int x, int* ok) {
    if (*R == NULL) {
        *ok = 0;
        return;
    } else if (x < (*R)->info) {
        ABB* esq = &(*R)->esq;
        Remove(esq, x, ok);
    } else if (x > (*R)->info) {
        ABB* dir = &(*R)->dir;
        Remove(dir, x, ok);
    } else {
        ABB aux = *R;
        *ok = 1;

        if ((*R)->esq == NULL && (*R)->dir == NULL) {
            free(aux);
            *R = NULL;
        } else if ((*R)->esq != NULL && (*R)->dir != NULL) {
            aux = (*R)->esq;

            while (aux->dir != NULL) {
                aux = aux->dir;
            }

            (*R)->info = aux->info;
            ABB* esq = &(*R)->esq;
            Remove(esq, (*R)->info, ok);
        } else {
            if ((*R)->esq == NULL) {
                *R = (*R)->dir;
            } else {
                *R = (*R)->esq;
            }
            free(aux);
        }
    }
}
```

Questão 2

(a) void BubbleSort (variável por referência V do tipo vetor de inteiros, variável N do tipo inteiro)
/* ordena o vetor V de tamanho N, pelo algoritmo BubbleSort, interrompendo o processo quando nenhuma troca for efetuada em uma “passada”. */

```c
void bubble_sort(int V[], int N) {
    int teve_troca;
    int k = N;
    int aux;

    for (int i = 0; i < N; i++) {
        teve_troca = 0;

        for (int j = 1; j < k; j++) {
            printf("\npassada %d comparou %d\n", i + 1, k);
            if (V[j - 1] > V[j]) {
                aux = V[j];
                V[j] = V[j - 1];
                V[j - 1] = aux;

                teve_troca = 1;
            }
        }

        k--; // Diminui o número de comparações a cada iteração

        // Se não teve troca é porque já está ordenado
        if (!teve_troca) {
            printf("\nnao teve troca na passada %d", i + 1);
            return;
        }
    }
}
```

(b) A organização inicial do vetor no pior caso é quando ele está ordenado na ordem inversa. No caso do exemplo dado seria V[6] = { 90, 47, 34, 29, 12, 7 }. O número de passos no pior caso é da ordem de $O(N^2)$.

(c) A organização inicial do vetor no melhor caso é quando ele já está ordenado. No caso do exemplo dado seria V[6] = { 7, 12, 29, 34, 47, 90 }. O número de passos no melhor caso é da ordem de O(N).

Questão 3

i) void CorrigeHeapSubindo (variável por referência Heap do tipo vetor de inteiros, variável LastPosition do tipo inteiro)
/* este procedimento corrige o Heap, posicionando o elemento que acabou de ser inserido em Heap[LastPosition] em sua posição adequada no heap/árvore/vetor. */

```c
void corrige_heap_subindo(int Heap[], int LastPosition) {
    int i;
    int aux;

    i = (LastPosition - 1) / 2; // índice do pai

    // se o LastPosition tem um pai e se seu valor é maior que o dele
    if (i >= 0 && Heap[LastPosition] > Heap[i]) {
        // troca os elementos do Heap
        aux = Heap[LastPosition];
        Heap[LastPosition] = Heap[i];
        Heap[i] = aux;

        corrige_heap_subindo(Heap, i); // corrige o pai
    }
}
```

ii) No pior caso ocorrerão log(N) trocas para corrigir o Heap.