TAB\_tpCondRet TAB\_DestruirTabuleiro(ptTabuleiro tabu) {

int i=0,j;

while(i<8) {

j=0;

while(j<8) {

if(tabu->tab[i][j].elemento != NULL) {

LIS\_DestroiLista(tabu->tab[i][j].ameacados);

LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

}

j++;

}

i++;

}

LIS\_DestroiLista(listaPecas);

return TAB\_CondRetOK;

}

Bloco A:

while(i<8) {

j=0;

while(j<8) {

if(tabu->tab[i][j].elemento != NULL) {

LIS\_DestroiLista(tabu->tab[i][j].ameacados);

LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

}

j++;

}

i++;

}

Bloco B:

while(j<8) {

if(tabu->tab[i][j].elemento != NULL) {

LIS\_DestroiLista(tabu->tab[i][j].ameacados);

LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

}

j++;

}

Bloco C:

if(tabu->tab[i][j].elemento != NULL) {

LIS\_DestroiLista(tabu->tab[i][j].ameacados);

LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

}

Bloco D:

LIS\_DestroiLista(tabu->tab[i][j].ameacados);

LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

Argumentação de Sequencia

AE: A função de destruir tabuleiro recebe um ponteiro para a struct TAG\_tabuleiro.

AS: Ao final da função o tabuleiro foi liberado da mémoria do computador, ou seja, destruído

A1: Declaro i e j, porém i aponta para a primeira linha

- int i=0,j ;

A1’:i aponta para o próxima linha do tabuleiro

i++;

A1’’: i aponta para a última linha

A2: j aponta para a primeira coluna

- j=0;

A2’: j aponta para a próxima coluna do tabuleiro

j++;

A2’’: j aponta para a última coluna

A3: LIS\_DestroiLista(tabu->tab[i][j].ameacados);

-Descrita na argumentação de seleção

A4: LIS\_DestroiLista(tabu->tab[i][j].ameacantes;

-Descrita na argumentação de seleção

A5: PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

-Descrita na argumentação de seleção

A6: LIS\_DestroiLista(listaPecas);

-AE: A função destrói lista recebe um ponteiro para uma de lista de peças

-AS: Ao terminar a função a lista de peças foi destruída, ou seja, excluída da memória do computador

A7: return TAB\_CondRetOK;

-Significa que a função acabou sem problemas

Argumentação de Seleção

if(tabu->tab[i][j].elemento != NULL)

AE(IF) && (C == T) + D => IF

IF

{

Executa A3: LIS\_DestroiLista(tabu->tab[i][j].ameacados);

-AE: A função de destruir lista recebe o endereço da lista de peças ameaçadas

-AS: A lista recebida por referência com as peças ameaçadas foi destruída, ou seja, apagada da memória do computador

Executa A4: LIS\_DestroiLista(tabu->tab[i][j].ameacantes);

-AE: A função de destruir lista recebe o endereço da lista de peças ameaçantes

-AS: A lista recebida por referência com as peças ameaçantes foi destruída, ou seja, apagada da memória do computador

Executa A5: PEC\_LiberaPeca((Peca\*)tabu->tab[i][j].elemento);

-AE: A função libera peça recebe o endereço da peça alocada na sua respectiva casa

-AS: A peça recebida por referência foi destruída, ou seja, liberada da memória do computador

}

AE(IF) && (C == F) => Beta 3.1

Argumentação de Repetição

Repetição Alpha: while(i<8) {…}

AE: A1

AS: A1’

Repetição Beta: while(j<8) {…}

AE: A2

AS: A2’

Alpha

Alpha 1: AE => AINV

-AINV diz respeito ao teste do descritor de estado i, é a assertiva de entrada da repetição Alpha

Alpha 2: AINV + (C == T) + BlocoA => BlocoB

-Significa que o descritor de estado está dentro dos limites do tabuleiro (linhas)

Alpha 3: AINV + (F == T) => Alpha 5

-Significa que A1 já percorreu todas as linhas e colunas do tabuleiro, ou seja, está fora dos limites do tabuleiro

Alpha 4: AE + (C == F) + BlocoA => AS

-Para a repetição, o que significa que todos os elementos do tabuleiro foram destruidos

Beta

Beta 1: AE => AINV2

-AINV diz respeito ao teste do descritor de estado j, é a assertiva de entrada da repetição Beta

Beta 2: AINV2 + (C == T) + BlocoC=> Beta 3

-Significa que o descritor de estado está dentro dos limites do tabuleiro (colunas)

Beta 3: Beta3 => IF

-Vai executar as linhas dentro do bloco IF

Beta 3.1: Beta 3.1 => A2’

-Atualiza o A2’ para a próxima coluna

Beta 4: AINV2 + (C == F) => A1’

-Significa que AINV2 já percorreu todas colunas do tabuleiro, ou seja, está fora dos limites do tabuleiro

-Além disso ativará o A1’

Beta 5: AE + (C == F) + Bloco B => AS

-Para a repetição, o que significa que todos os elementos a linha AINV foram destruidos