Corretude Gamão

Bloco_1:

```
int Jogo_CondicaoFinalizar(JOGO_tppJogo pJogo, PECAFINALIZADA_tppLista PecFin_jogador, char
player)
{
AE->
	char cor;
	int i=0,qtd=0,soma=0;
AI1->
	TAB_IrInicioTabuleiro(pJogo->Tabuleiro);
AI2->
	if(player=='p')
		TAB_AvancarElementoCorrente(pJogo->Tabuleiro,18);
AI3->
	while(i<=5)
	{
		TAB_ObterCorNo(pJogo->Tabuleiro,&cor);
		if(cor==player)
		{
			QtdElemento((LIS_tppLista)LIS_ObterValor(pJogo->Tabuleiro),&qtd);
			soma+=qtd;
		}
		TAB_AvancarElementoCorrente(pJogo->Tabuleiro,1);
		i++;
	}
AI4->
	QtdElemento(PecFin_jogador,&qtd);
AI5->
	soma+=qtd;
AI6->
	if(soma==CONDFINAL)
		return TRUE;
	else
		return FALSE;
AS->
}
```

Argumentação Sequencial:

AE: - Existem as estruturas pJogo e PecaFin_Jogador.
- Há peças a serem verificadas.
- Ponteiro corrente pode estar apontando para a posição 0 ou 18 da lista de peças do Tabuleiro.

AS: - Retorna TRUE caso a condição de ter todas as peças de um jogador no último quadrante, retorna FALSE caso contrário.

AI1: - Variáveis 'i', 'qtd' e 'soma' são declarados e definidos com valor 0.

AI2: - Ponteiro corrente do tabuleiro aponta para o início do tabuleiro.

AI3: - Se a variável 'player' tiver 'p' como atribuição, o ponteiro corrente do tabuleiro aponta para a posição 18, que seria o último quadrante do tabuleiro. Caso não tenha essa atribuição, nada é feito com o ponteiro.

AI4: - A variável 'soma' vai conter a quantidade de elementos do jogador naquele quadrante.

AI5: - Armazena na variável 'qtd' a quantidade de peças finalizadas.

AI6: - A variável 'soma' vai conter a quantidade de elementos do jogador naquele quadrante mais a quantidade de peças finalizadas pelo jogador. Faz-se uma verificação de soma com CONDFINAL, que possui valor padrão 15.

Bloco_2:

```
if(player=='p')
       TAB_AvancarElementoCorrente(pJogo->Tabuleiro,18);     <-B1
```

Argumentação de Seleção:

AE: - Considera o ponteiro corrente do Tabuleiro apontando para o início.

AS: - Avança o ponteiro corrente do Tabuleiro para a posição 18 ou mantém sua posição.

1) AE && (C==T) + B1 => AS
pela AE, o ponteiro corrente aponta para o início do Tabuleiro, como (C==T), então ponteiro corrente começa a apontar para a posição 18 do Tabuleiro, valendo AS.

2) AE && (C==F) => AS
pela AE, o ponteiro corrente aponta para o início do Tabuleiro, como (C==F), então ponteiro corrente continua apontando para o início do Tabuleiro, valendo AS.

Bloco_3:

```
while(i<=5)
{
        TAB_ObterCorNo(pJogo->Tabuleiro,&cor);
        if(cor==player)
        {
                QtdElemento((LIS_tppLista)LIS_ObterValor(pJogo->Tabuleiro),&qtd);
                soma+=qtd;
        }
        TAB_AvancarElementoCorrente(pJogo->Tabuleiro,1);
        i++;
}
```

-B

Argumentação de Repetição

AE: - AI3

AS: - AI4

AINV: - Existem dois conjuntos a contar e ja contado
- Ponteiro corrente aponta para o elemento do a contar

1) AE => AINV
- Pela AE, ponteiro corrente do Tabuleiro está no início do tabuleiro, ou na posição 18. Caso ele esteja no início os cinco próximos elementos estão no conjunto a contar, e o conjunto ja contado está vazio. Caso ele esteja na posição 18 os cinco próximos elementos estão no conjunto a contar, e o conjunto ja contado contém todo o resto do Tabuleiro.

2) AE && (C==F) => AS
- Pela AE, ponteiro corrente do Tabuleiro está no início do tabuleiro, ou na posição 18, o ponteiro chegou na 5º posição depois do início ou da posição 18.

3) AE && (C==T) + B => AINV
- Para (C==T), primeira posição é verificada. Neste caso ela passa do conjunto a contar para ja contado e ponteiro corrente do Tabuleiro é reposicionado. Vale AINV.

4) AINV && (C==T) + B => AINV
- Para garantir que AINV seja válida a cada ciclo, B garante que uma peça passe do conjunto a contar para ja contado e ponteiro corrente do Tabuleiro seja reposicionado.

5) AINV && (C==F) => AS
- Pela AE, ponteiro corrente do Tabuleiro está no início do tabuleiro, ou na posição 18, o ponteiro chegou na 5º posição depois do início ou da posição 18.

6) Término
- A cada ciclo um elemento do Tabuleiro é retirado do conjunto a contar. Como esta conjunto possui um número finito de elementos, a repetição terminará em um número finito de passos.

Bloco_4:

```
if(soma==CONDFINAL)
	return TRUE;	<-B1
else
	return FALSE;	<-B2
```

Argumentação de Seleção

AE: - AI6

AS: - Retorna TRUE caso 'soma' possui o mesmo valor que CONDFINAL, retorna FALSE caso contrário.

1) AE && (C==T) + B1 => AS

-Pela AE 'soma' pode ter o mesmo valor que CONDFINAL, como (C=T), então soma=CONDFINAL. Ao executar B1, a função retorna TRUE, valendo a AS.

2) AE && (C==F) + B2 => AS

-Pela AE 'soma' pode não ter o mesmo valor que CONDFINAL, como (C=F), então soma!=CONDFINAL. Ao executar B2, a função retorna FALSE, valendo a AS.

Bloco_5:

```
AE->
	TAB_ObterCorNo(pJogo->Tabuleiro,&cor);
AI1->
	if(cor==player)
	{
		QtdElemento((LIS_tppLista)LIS_ObterValor(pJogo->Tabuleiro),&qtd);
		soma+=qtd;
	}
AI2->
	TAB_AvancarElementoCorrente(pJogo->Tabuleiro,1);
AI3->
	i++;
AS->
```

Argumentação Sequencial:

AE: - Tabuleiro existe, possui peças e o ponteiro corrente do Tabuleiro está apontando para a posição inicial ou para a posição 18 do Tabuleiro.

AS: - A variável 'soma' vai conter a quantidade de elementos do jogador naquele quadrante.
- A variável 'i' é acrescentada de 1.

AI1: - A variável cor recebe qual cor da peça está na posição que o corrente do Tabuleiro aponta.

AI2: - Se a variável 'cor' tiver a o mesmo conteúdo da variável 'player', então 'qtd' recebe o valor referente a quantidade de elementos da posição apontada pelo ponteiro corrente do Tabuleiro.

AI3: - Ponteiro corrente do Tabuleiro avança em uma posição.

Bloco_6:

```
AE->
       if(cor==player)
       {
              ____________________________________________________________
              QtdElemento((LIS_tppLista)LIS_ObterValor(pJogo->Tabuleiro),&qtd);   |-B
              soma+=qtd;__________________________________________________________|
       }
AS->
```

Argumentação de Seleção:

AE: - A variável 'cor' recebe qual cor da peça está na posição que o corrente do Tabuleiro aponta.

AS: - Se a variável 'cor' tiver o mesmo conteúdo da variável 'player', então 'soma' adiciona ao seu valor a quantidade de peças que o ponteiro corrente do Tabuleiro aponta. Caso contrário, nada é feito.

1) AE && (C==T) + B => AS
-Pela AE 'cor' pode ter o mesmo valor que 'player', como (C=T), então cor=player. Ao executar B, ao seu valor a quantidade de peças que o ponteiro corrente do Tabuleiro aponta, valendo a AS.

2) AE && (C==F) => AS
-Pela AE 'cor' pode não ter o mesmo valor que 'player', como (C=F), então soma!=player. Nada é feito, valendo a AS.

Bloco_7:

```
AE ->
       QtdElemento((LIS_tppLista)LIS_ObterValor(pJogo->Tabuleiro),&qtd);
AI1->
       soma+=qtd;
AS ->
```

Argumentação Sequencial:

AE: - Existe a estruturas pJogo, variável soma e variável qtd.
- Ponteiro corrente pode estar apontando para uma posição da lista de peças do Tabuleiro.

AS: - Variável 'soma' adiciona ao seu valor a quantidade de peças que o ponteiro corrente do Tabuleiro aponta.

AI1: - Variável 'qtd' recebe a quantidade de peças que o ponteiro corrente do Tabuleiro aponta.