GCC255 - REO 2: Primeira Lista de Exercícios

Mateus Carvalho Gonçalves - 201810245
Otávio de Lima Soares - 201811022
Pedro Antônio de Souza - 201810557

Exercício 1)

a) Consideramos o número de entradas como uma condição de entrada por falta de especificações no comando, já que um programa real deve sempre tratar essa quantidade corretamente.

Condição de entrada	Classes válidas	Classes inválidas
Número de entradas (N)	N === 3	N != 3
Valores das entradas (V)	V >= 1	V < 1
Condição (C) de formação de triângulo (soma de dois deve ser maior que o terceiro)	Para entradas V_0, V_1, V_2 => $V_0 + V_1 > V_2$ && $V_0 + V_2 > V_1$ && $V_1 + V_2 > V_0$	Para entradas V_0, V_1, V_2 => $V_0 + V_1 <= V_2$ && $V_0 + V_2 <= V_1$ && $V_1 + V_2 <= V_0$

b) Nos casos de teste abaixo não foram usados valores que podem ser usados na Análise do Valor Limite, já que é um item posterior do exercício. Também, a entrada é representada apenas pelos valores V, os valores das outras condições são implícitas.

T = {
(<5, 6, 7>, escaleno),
(<100, 100, 150>, isósceles),
(<10, 10, 10>, equilátero),
(<22>, N inválido),
(<-5, 3, 4>, V inválido),
(<3, 7, 12>, C inválido)
}

c) Para N os valores de teste baseados nos limites são {2, 3, 4}.
Para V os valores de teste baseados nos limites são {0, 1}.
Para C, considerando quaisquer ordens de V_0, V_1 e V_2, os limites são
{
$V_0 + V_1 < V_2$, sendo $V_2 === V_0 + V_1 - 1$;
$V_0 + V_1 === V_2$;
$V_0 + V_1 > V_2$, sendo $V_2 === V_0 + V_1 + 1$;
}.

d) T = {
(<2, 2>, N inválido),
(<2, 2, 2>, equilátero),

(<2, 2, 2, 2>, N inválido),
(<0, 100, 150>, V inválido),
(<1, 10, 10>, isósceles),
(<3, 7, 9>, escaleno),
(<3, 7, 10>, C inválido),
(<3, 7, 11>, C inválido)
}

Exercício 2)

a) Para o método **search()**, deve-se gerar classes de equivalência considerando a nulidade ou existência da lista e do elemento entrado. Porém, os elementos válidos possuem duas classes: pertencente à lista e não pertencente. A tabela abaixo ilustra as classes.

Condição de entrada	Classes válidas	Classes inválidas
Lista de elementos (L)	L != NULL	L == NULL
Elemento procurado (E)	Pertence Não pertence	E == NULL

b) A partir da interpretação do primeiro exemplo de execução presente no código do enunciado, foi considerado que, nos casos em que o elemento ocorre mais de uma vez na lista, espera-se que o programa retorne qualquer um dos índices de sua ocorrência. Assim, os casos de teste abaixo foram gerados:

T = {
(<NULL, 1>, L inválido),
(<[1, 2, 3], NULL>, E inválido),
(<[0, 16, 4, 20], 4>, 2),
(<[13, 27, 8], 7>, -1),
(<[6, 0, 6, 6], 6>, 0 or 2 or 3)
}

Exercício 3)

O programa possui 4 possibilidades de execução: a possibilidade em que o número é divisível por 3, a que o número é divisível por 5, a que o número é divisível por 3 e por 5 ao mesmo tempo e por último a que o número não é divisível nem por 3 e nem por 5.

Desta forma podemos agrupar as entradas T1, T2, T3 e T4 no grupo dos divisíveis por 3. As entradas T1, T2 e T5 no grupo dos divisíveis por 5. T1 e T2 no grupo dos divisíveis por ambos e não haverá entrada para o grupo dos não divisíveis por ambos.

Logo, pelos conceitos de particionamento de equivalência e valor limite podemos excluir as entradas T2 e T3 que estariam gerando casos de teste repetitivos. Porém as entradas fornecidas não seriam suficientes para testar todas as características do programa, pois não há nenhuma entrada classificada no grupo dos não divisíveis nem por 3 e nem por 5.