GCC255 - REO 2

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

Programa 1)

----------Programa 1----------

```
public int findLast(int[] x, int y){
//effects: if x == null throw
NullPointerException
//else return the index of the last
element
//int x that equals y.
//if no such element exists, return -1
        for(int i=x.length-1; i>0; i--)
        {
                if(x[i] == y)
                {
                                return i;
                }
        }
        return -1;
}
// test: x=[2,3,5]; y=2
// expect = 0
```

Defeito do programa:

A linha `for(int i=x.length-1; i>0; i--)` deve ser corrigida para `for(int i=x.length-1; i>=0; i--)`.

1. Caso de teste 1:
    - t1 = ([2,3,5], 3, 1) //(vetor x, int y, saída esperada do programa)
        - Para findLast([2,3,5], 3) a saída será correta, valor 1.
2. Caso de teste 2:
    - t2 = ([5,3, 2], 5, 0)
        - Para findLast([5,3,2], 3) a saída será incorreta, valor -1.

Levando em consideração os dois casos de teste acima , o defeito é executado apenas em t2, gerando um erro também apenas para o caso 2 pois um estado incorreto é atingido (ou seja, o vetor termina de ser lido na posição 1 ao invés da posição zero), por isso t2 resulta em uma falha no programa e t1 executa com resposta esperada, dando uma falsa sensação de que o programa está correto.

Esse estado de erro mencionado no parágrafo anterior dá-se ao registro do valores das variáveis x, i, y e o contador do programa PC, que significa a linha em que está sendo executada naquele momento. Considerando a execução de t2, o estado do comando if na última iteração do loop é (x = [5,3,2], i = 1, y = 5, PC=if). Observe que este estado está em erro porque o valor de i deveria ser 0 e não 1 na última iteração. Desta maneira o programa resulta em saída -1 por não verificar a posição 0 do vetor que contém o valor 5 que é igual ao valor de y. Em t1, devido ao fato de 3 (que é o valor de y) estar situado na posição 1 do vetor, o programa gera a saída esperada, dando uma falsa sensação de que o programa está correto.

Desta forma, se corrigirmos a linha `for(int i=x.length-1; i>0; i--)` para `for(int i=x.length-1; i>=0; i--)` conseguimos um resultado correto tanto em t2 quanto no caso de teste dado junto ao programa.

Programa 2)

```
--------------Programa 2---------------

public static int lastZero(int[] x){
//effects: if x == null throw
NullPointerException
//else return the index of the last 0
in x.
//return -1 if 0 does not occur in x
        for(int i=0; i<x.length; i++)
        {
                if(x[i] == 0)
                {
                                return i;
                }
        }
        return -1;
}

// test: x=[0,1,0]
// expect = 2
```

Defeito do programa:

A linha `for(int i=0; i<x.length; i++)` deveria ser `for(int i=x.length-1; i>=0; i--)`

1. Caso de teste 1:
    - t1 = ([0,1,1], 0) //(vetor x, saída esperado do programa)
        - para lastZero([0,1,1]) a saída será correta, valor 0.

Para o caso de teste acima o defeito é executado, gerando um estado de erro pois para que ele encontre a última posição do número 0 no vetor é preciso que ele comece a leitura do vetor de trás para frente. Esse caso de teste não resulta em falha pois há apenas um número 0 no conteúdo do vetor, por isso o resultado para este caso será correto, gerando falsa sensação de eficácia no programa.

Desta forma se corrigirmos a linha `for(int i=0; i<x.length; i++)` por `for(int i=x.length-1; i>=0; i--)` conseguimos um resultado correto tanto em t1 quanto no caso de teste dado junto ao programa.

Programa 3)

```
--------------Programa 3---------------

public int countPositive(int[] x){
 //effects: if x == null throw
NullPointerException
//else return the number of
//positive elements in x.
        int count = 0;
        for(int i=0; i<x.length; i++)
        {
                if(x[i] >= 0)
                {
                                count++;
                }
        }
        return count;
}
// test: x=[-4,2,0,2]
// expect = 2
```

Defeito do programa:

Considerando o caso de teste especificado nas duas últimas linhas, o programa não deve considerar 0 como número positivo. Dessa forma, a condição `if(x[i] >= 0)` deveria ser `if(x[i] > 0)`.

1. Caso de teste 1:
    - t1 = ([0, 3, 0], 1) // (vetor x, saída esperada do programa)
        - Para countPositive([0, 3, 0]) a saída será incorreta, valor 3.
2. Caso de teste 2:
    - t2 = ([5, 3, 2], 3)
        - Para countPositive([5,3,2]) a saída será correta, valor 3.
3. Caso de teste 3:
    - t3 = ([], 0)
        - Para countPositive([]) a saída será correta, valor 0.

Considerando os casos de teste acima, o defeito é executado em t1 e t2, logo, há um erro nessas execuções, e em t3, como `x.length == 0`, não entra no escopo do for e o defeito não é executado. Porém, apenas em t1 o erro de domínio gera falha, visto que a condição é computada como verdadeira para 0 também, gerando um valor de saída diferente do esperado.

Ao realizar a correção apontada no primeiro parágrafo da seção, todos os casos de teste executam corretamente.

Programa 4)

```
--------------Programa 4---------------

public static int oddOrPos(int[] x){
 //effects: if x == null throw
NullPointerException
//else return the number of elements in
x that
//are either odd or positive (or both).
        int count = 0;
        for(int i=0; i<x.length; i++)
        {
                if(x[i]%2 == 1 || x[i]>0)
                {
                                count++;
                }
        }
        return count;
}
// test: x=[-3,-2,0,1,4]
// expect = 3
```

Defeito do programa:

Pode-se inferir pelo caso de teste descrito nas últimas linhas do código que deve-se considerar o número zero como positivo. Porém, ao executar o caso de teste dado, o resultado obtido é 2, diferentemente do que era esperado. Isso ocorre pois há um defeito de comparação na linha `if(x[i]%2 == 1 || x[i]>0)`. Esse defeito pode ser corrigido substituindo a linha por `if(x[i]%2 == 1 || x[i]>=0)`.

1. Caso de teste 1:
    - t1 = ([], 0) // (vetor x, saída esperada do programa)
        - Para oddOrPos([]) a saída será correta, ou seja, valor 0.
2. Caso de teste 2:
    - t2 = ([1, -2, 3, 4], 3)
        - Para oddOrPos([1,-2,3,4]) a saída será correta, ou seja, valor 3.

Em t1 temos que `x=[]`, portanto, `x.length == 0`. Assim, não ocorrerá nenhuma execução do laço de repetição que contém o defeito. Por consequência, a função retornará o valor `0` como esperado, gerando falsa sensação de eficácia do programa.

Essa sensação também é gerada em t2. Nesse caso, apesar do defeito ser executado, o valor retornado é `3` como esperado. Isso ocorre pois o erro está na condição para o valor `0`. Assim, qualquer vetor `x` que não contenha o valor `0`, executará o defeito mas não resultará em um erro.

Ao realizar a correção da linha `if(x[i]%2 == 1 || x[i]>0)` por `if(x[i]%2 == 1 || x[i]>=0)`, obtém-se o resultado correto tanto nos dois casos de teste descritos quanto no caso de teste dado junto ao programa.