

Nome: _____

Defeito, Erro e Falha

Abaixo são apresentados quatro programas com defeito. Para cada programa é ilustrado um caso de teste que resulta em uma falha. Responda as questões abaixo.

```
public int findLast (int[] x, int y) {
//Effects: If x==null throw NullPointerException
// else return the index of the last element
// in 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
// Expected = 0
```

```
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]
// Expected = 2
```

```
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]
// Expected = 2
```

```
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]
// Expected = 3
```

- 1) Identifique o defeito de cada programa.
- 2) Se possível, identifique um caso de teste que não executa o defeito. Explique
- 3) Se possível, identifique um caso de teste que executa o defeito, mas não resulta em um erro. Explique
- 4) Se possível, identifique um caso de teste que resulta em um erro, mas não em uma falha. Explique
- 5) Corrija o defeito e verifique se o caso de teste dado produz a saída esperada.