Pruebas Unitarias Simulación Turing Machine

1. Escenarios

|  |  |  |
| --- | --- | --- |
| Nombre | Clase | Escenario |
| Setup1() | Letter | ‘A’ |
| Setup2() | Letter | ‘A’ <–> ‘B’ |
| Setup3() | Letter | ‘Z’ <–> ‘A’ <–> ‘B’ |

|  |  |  |
| --- | --- | --- |
| Nombre | Clase | Escenario |
| Setup1() | Turing Machine | vacío |
| Setup2() | Turing Machine | ‘A’ |
| Setup3() | Turing Machine | ‘A’ <–> ‘B’ |
| Setup4() | Turing Machine | ‘A’ <–> ‘B’ <–> ‘C’ |
| Setup5() | Turing Machine | ‘A’ <–> ‘B’ <–> ‘C’ <-> ‘D’ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba # 1 | Verificar los getters y los setters de la clase Letter | | | |
| Clase | Método | Escenario | Valores de  Entrada | Resultado |
| Letter | \*getC() | Setup1() | *Ninguno* | \*‘A’ |
| \*A.getNextLetter.getC()  \*B.getBackLetter.getC() | Setup2() | *Ninguno* | \*‘B’  \*’A’ |
|  | \*Z.getNextLetter.getC()  \*A.getBackLetter.getC() | Setup3() | *Ninguno* | \*’A’  \*’Z’ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba # 1 | Verificar imprima la letra correcta al tratar de imprimir la letra que lee una cabeza. Si la cabeza no lee nada, deberá imprimir #. | | | |
| Clase | Método | Escenario | Valores de  Entrada | Resultado |
| Turing  Machine | \*showLetter(); | Setup1() | *‘0’*  *‘1’*  *‘2’* | “#”  En los tres subcasos. |
|  | \*showLetter(); | Setup4() | *‘0’*  *‘1’*  *‘2’* | “A”  “B”  “C” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba # 2 | Verificar que los elementos se agreguen correctamente a la lista. El elemento deberá ocupar la posición de la cabeza donde fue agregado. Y las demás cabezas deberán actualizar su posición. | | | |
| Clase | Método | Escenario | Valores de  Entrada | Resultado |
| Turing  Machine | \*addLetter(); | Setup1() | *‘0’ , ‘A’*  *‘1’ , ‘A’*  *‘2’ , ‘A’*  *(Por aparte)* | C0=’A’, C1=’A’, C2=’A’  En los tres subcasos. |
| \*addLetter(); | Setup2() | *‘0’ , ‘B’*  *‘1’ , ‘B’*  *‘2’ , ‘B’* | C0=’B’, C1=’B’, C2=’A’  C0=’B’, C1=’B’, C2=’A’  C0=’A’, C1=’A’, C2=’B’ |
|  | \*addLetter(); | Setup3() | *‘0’ , ‘C’*  *‘1’ , ‘C’*  *‘2’ , ‘C’* | C0=’C’, C1=’A’, C2=’B’  C0=’A’, C1=’C’, C2=’B’  C0=’A’, C1=’B’, C2=’C’ |
|  | \*addLetter(); | Setup4() | *‘0’ , ‘D’*  *‘1’ , ‘D’*  *‘2’ , ‘D’* | C0=’D’, C1=’A’, C2=’C’  C0=’A’, C1=’D’, C2=’C’  C0=’A’, C1=’B’, C2=’D’ |
|  | \*addLetter(); | Setup5() | *‘0’ , ‘E’*  *‘1’ , ‘E’*  *‘2’ , ‘E’* | C0=’E’, C1=’B’, C2=’D’  C0=’A’, C1=’E’, C2=’D’  C0=’A’, C1=’C’, C2=’E’ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba # 3 | Verificar que los elementos se eliminen correctamente de la lista. Y que las demás cabezas actualicen su posición. | | | |
| Clase | Método | Escenario | Valores de  Entrada | Resultado |
| Turing  Machine | \*removeLetter(); | Setup1() | *‘0’*  *‘1’*  *‘2’*  *(Por aparte)* | C0=null, C1=null, C2=null  En los tres subcasos. |
| \*removeLetter(); | Setup2() | *‘0’*  *‘1’*  *‘2’* | C0=null, C1=null, C2=null |
|  | \*removeLetter(); | Setup3() | *‘0’*  *‘1’*  *‘2’* | C0=’B’, C1=’B’, C2=’B’  C0=’B’, C1=’B’, C2=’B’  C0=’A’, C1=’A’, C2=’A’ |
|  | \*removeLetter(); | Setup4() | *‘0’*  *‘1’*  *‘2’* | C0=’B’, C1=’B’, C2=’C’  C0=’A’, C1=’A’, C2=’C’  C0=’A’, C1=’A’, C2=’B’ |
|  | \*removeLetter(); | Setup5() | *‘0’*  *‘1’*  *‘2’* | C0=’B’, C1=’C’, C2=’D’  C0=’A’, C1=’C’, C2=’D’  C0=’A’, C1=’B’, C2=’C’ |