**Diseño de pruebas**

|  |  |
| --- | --- |
| Prueba No: | 1 |
| Tipo de Prueba: | Unitaria Automática |
| Método a Probar: | InsertarVertice(Vértice k) |
| Entradas: | Elemento de tipo vértice nuevo a insertar en grafo |
| Salidas: | Agrega un vértice con las características del elemento pasado como parámetro |

|  |  |
| --- | --- |
| Caso de Prueba No: | 1.1 |
| Objetivo: | Probar que el método agrega correctamente el nuevo elemento al grafo cuando el grafo se encuentra vacío |
| Escenario: | * Escenario 1 |
| Entrada(s): | * “A” |
| Salida Esperada: | Grafo con 1 elemento (Vértice). |

|  |  |
| --- | --- |
| Caso de Prueba No: | 1.2 |
| Objetivo: | Probar que el método agrega correctamente el nuevo elemento al grafo cuando ya hay otros elementos |
| Escenario: | * Escenario2 |
| Entrada(s): | * “C” |
| Salida Esperada: | Grafo con 3 elementos |

|  |  |
| --- | --- |
| Caso de Prueba No: | 1.3 |
| Objetivo: | Probar que el método agrega correctamente el nuevo elemento al grafo cuando ya hay conexiones entre más de dos vértices |
| Escenario: | * Escenario 3 |
| Entrada(s): | * “C” |
| Salida Esperada: | Grafo con 3 elementos y dos de ellos conectados mediante una arista |

|  |  |
| --- | --- |
| Caso de Prueba No: | 1.4 |
| Objetivo: | Probar que el método no agrega correctamente el nuevo elemento al grafo cuando este ya existe |
| Escenario: | * Escenario 3 |
| Entrada(s): | * “A” |
| Salida Esperada: | Excepción indicando que este vértice ya existe |

|  |  |
| --- | --- |
| Prueba No: | 2 |
| Tipo de Prueba: | Unitaria Automática |
| Método a Probar: | Crear Arista(Vertice1, Vertice2, peso) |
| Entradas: | Vertice1, vertice2 y el peso de la arista |
| Salidas: | Un grafo en el cual esos dos vértices están conectados por una arista de ese peso |

|  |  |
| --- | --- |
| Caso de Prueba No: | 2.1 |
| Objetivo: | Probar que el método genera la arista |
| Escenario: | Escenario 2 |
| Entrada(s): | * “A”,”B”, 3 |
| Salida Esperada: | Escenario 3 |

|  |  |
| --- | --- |
| Caso de Prueba No: | 2.2 |
| Objetivo: | Probar que el método retorna una excepción cuando alguno de los vértices no exista en el grafo |
| Escenario: | * Escenario1 |
| Entrada(s): | * “A”,”B”, 3 |
| Salida Esperada: | Excepción |

|  |  |
| --- | --- |
| Prueba No: | 3 |
| Tipo de Prueba: | Unitaria Automática |
| Método a Probar: | BFS(Vértice K) |
| Entradas: | Nodo en donde comenzará a hacer el árbol por anchura |
| Salidas: | Un árbol que muestra el recorrido y un entero indicando el total de nodos recorridos |

|  |  |
| --- | --- |
| Caso de Prueba No: | 3.1 |
| Objetivo: | Probar que el método busca el árbol resultante |
| Escenario: | Ciudad1   * String Nombre = “Cali” * int identificador = ” 1”   Ciudad2   * String Nombre = “Medellín” * int identificador = ” 2”   Ciudad3   * String Nombre = “Bogotá” * int identificador = ” 3” |
| Entrada(s): | int identificador = “1” |
| Salida Esperada: | Un ArrayList que contenga las conexiones de la ciudad 1 |

|  |  |
| --- | --- |
| Prueba No: | 4 |
| Tipo de Prueba: | Unitaria Automática |
| Método a Probar: | EliminarVertice(Vertice k) |
| Entradas: | El elemento al cual se va a borrar |
| Salidas: | Grafo donde ya no pertenezca ese vértice |

|  |  |
| --- | --- |
| Caso de Prueba No: | 4.1 |
| Objetivo: | Verificar que el método lanza excepción cuando el vértice no existe |
| Escenario: | * Escenario4 |
| Entrada(s): | * “Z” |
| Salida Esperada: | Excepción indicando que el grafo no tiene ese vértice |

|  |  |
| --- | --- |
| Caso de Prueba No: | 4.2 |
| Objetivo: | Probar que el funciona correctamente |
| Escenario: | * Escenario4 |
| Entrada(s): | * “A” |
| Salida Esperada: | Un Grafo donde ya no pertenezca A |