ICT 283 Assignments UAT

Class : Date class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Description (include why the test is needed) | Actual Test Data | Expected Output | Pass/Fail |
| 1 | Mutator for Day  (SetDay) Requires testing as it is part of the operation of setting values. | Input Value “23” | Expected value 23 . | Pass |
| 2 | Mutator for Month (SetMonth)  Requires testing as it is part of the operation of setting values. | Input Value  “12” | Expected value 12. | Pass |
| 3 | Mutator for Year (SetYear) Requires testing as it is part of the operation of setting values. | Input Value “1994” | Expected value 1994 | Pass |
| 4 | Assessor for Day  (GetDay) Requires testing as it is part of the operation of getting values. | Input Value “23” | Expected value 23. | Pass |
| 5 | Assessor for month  (GetMonth)  Requires testing as it is part of the operation of getting values. | Input Value “12” | Expected value 12. | Pass |
| 6 | Assessor for year (GetYear)  Requires testing as it is part of the operation of getting values. | Input Value  “1994” | Expected Value  1994. | Pass |
| 7 | Test Constructor  Testing is required as it is needed during constructing of object with assigning values via parameter | Input value 26 for day, 10 for month, 1995 for year | Expected value  26 day, 10-month, 1995 year. | Pass |

Class: Time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Description (include why the test is needed) | Actual Test Data | Expected Output | Pass/Fail |
| 1 | Mutator for time  (SetTime)  Requires testing as it is part of the operation of setting values. | Input value  “9:00” | Output value  “9:00” | Pass |
| 2 | Assessor for time  (GetTime)  Requires testing as it is part of the operation of getting values. | Input value  “9:00” | Output Value  “9:00” | Pass |
| 3 | Constructor  Requires as it is part of the construction of class object through initializing parameters | Input value  “12:00” | Output Value  “12:00” | Pass |

Class: WindSpeed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Description (include why the test is needed) | Actual Test Data | Expected Output | Pass/Fail |
| 1 | Mutator for windspeed  SetWindSpeed | Input value  “203” | Output value  “203” | Pass |
| 2 | Assessor for time  (GetWindSpeed)  Requires testing as it is part of the operation of getting values. | Input value  “203” | Output Value  “203” | Pass |
| 3 | Constructor  Requires as it is part of the construction of class object through initializing parameters | Input value  “12:00” | Output Value  “12:00” | Pass |

Test Template Vector

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Description (include why the test is needed) | Actual Test Data | Expected Output | Pass/Fail |
| 1 | Creating Vector with “Int” as variable type.  Requires testing as we have used T for defining any declared variables should be allowed | Create the vector object with “int” declared | Created the vector object with int declared, Without any errors | Pass |
| 2 | Creating Vector with “String” as variable type.  Requires testing as we have used T for defining any declared variables should be allowed | Create the vector object with “string” declared | Created the vector object with “string” declared, without any errors | Pass |
| 3 | Creating an int vector and use (add(variable))  Required for testing as users are required to add values into vector. | Using the vector that has been created for test 1 ) we have used a for loop add values from 0 – 7 into vector. | Expected output is 0 – 7 and will be loaded into a specific index in the vector. | Pass |
| 4 | Getting the size value of the vector in test 3.  Requires as this is a minimal feature of vector. | Using the command size() we should be able to get the number of elements currently in the vector, since previously we have loaded 0 – 7. | Expected output to be 8 elements. | Pass |
| 5 | Getting the capacity value of the vector in test 3.  Requires as this is a minimal feature of vector | Default capacity of 8. | Since the default capacity of vector should be 8, we should be able to store 8 elements without resizing. Hence, we expect 8 for capacity | Pass |
| 6 | Testing operator “ [index ] “ operator to call index in array and return stored value.  Requires as this is operator is important to call the value in the assigned dynamic array in our vector template | Using the operator “[ index]”  We have called index 0. The value we have added into index 0 is 0. | The expected value from the calling operator is 0. | Pass |
| 7 | Testing InsertAt()  Insert at behaves as the first parameter is the index and second is the value. It is required as it is part of the minimal vector design | Using the method. We have input (3,500) 3 being index and 500 being value. | We expect that in vector[3], value to be 500. | pass |
| 8 | Testing Deletefrom(index1, index2)  Deletefrom behaves to delete information from first index and second index . This test is required as it is part of the vector feature. | Using method deletefrom(3,5) 3 index to 5 index . | we will expect that index 3 to index 5 values will be deleted and the vector to resize . Based on the previous example we added 500 into 3 index. we expect value 2,500,3 to be deleted | Pass |
| 9 | Testing constructor overloading vector <int> test (10)  Should assign vector with 10 element size vector.  This test is required as it is the minimal structure of a vector. | Using this constructor we initialize vector element size of 10. | Since we know that the capacity is always 2 times the element size initalizing it wil 10 will cause the capacity to be 20 default size.  Expected capacity is 20 and vector element size 10. | Pass |
| 10 | Testing assignment operater “=”. Deep copying is expected and also this test is required as it is included in the minimal structure of vector. | Using assignment operator we have created another vector with default values element size 0 and capacity 8. using = from test 9 vector which has 20 capacity size size and 10 element size . | We expect that since we copy the new vector with default size to test 9 vector .  We expect that the test 9 vector will hold the new vector default values of 0 size element and 8 default capacity. | pass |
| 11 | Testing Remove() this will remove everything and reset the vector to default settings.  This test is needed as it is part of the minimal structure of vector. | Applying remove() on vector. | Using the previous vector from test 10 we expect the same values as output which is size 0 and default size of 8 | pass |
| 12 | Testing modifyat(index, value)  Modify at modifies the element at index specified by user. This test is required as it is part of the minimal structure | Applying modifyat(5,50)  Index 5 and place in value 50. | Expected outcome when called index 5 , is 50. | pass |
| 13 | Testing empty()  Empty returns boolean. True/ false .  If there is no elements in vector it will return true.  This test is required as it is part of the minimal structure. | Applying empty()  On the vetore we have applied this vector in a if statement on the vector used for test 12. If the vector is not empty It will print out implementation successful for empty() | Since we have added 50 at index 5.    The program should return false as there is an element in the array. and print implementation successful for empty() | pass |