**Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected result | Actual results | Reason for the chosen example |
| Test for copy constructor | | | |
| Set copyS(s); | copyS to be { 1 3 5 7 } | { 1 3 5 7 } | To test copy constructor function |
| Print copyS after copyS.insert(8); | copyS to be { 1 3 5 7 8 } | { 1 3 5 7 8 } | Inserting new value to copyS to see copyS is individual set |
| Print s | s to be { 1 3 5 7 } | { 1 3 5 7 } | s to be different from copyS |
| Set copySet2\_a(set2); | opySet2\_a to be { } | { } | To check copying empty Set using copy constructor |
| Test for assignment operator | | | |
| copyT = t; | copyT to be { 3 4 5 } | { 3 4 5 } | Copying set using assignment operator |
| Print t  Print copyT  After copyT.insert(6); | t tobe { 3 4 5 }  copyT to be { 3 4 5 6 } | { 3 4 5 }  { 3 4 5 6 } | Inserting new value to copyT to see copyT is individual set and t to be different from copyT |
| copySet2\_b = set2; | copySet2\_b to be { } | { } | To check copying empty Set using assignment operator |
| Test for overloading operator< (Bonus) | | | |
| Print setP  Print setQ  setP < setQ | { 2 3 }  { 2 3 5 }  true | { 2 3 }  { 2 3 5 }  true | To check setP is proper subset of SetQ |
| Change the values of setP and setQ  Print setP  Print setQ  setP < setQ | { 2 3 5 }  { 2 3 5 }  false | { 2 3 5 }  { 2 3 5 }  false | To check setP is NOT proper subset of SetQ |
| Change the values of setP and setQ  Print setP  Print setQ  setP < setQ | { 2 }  { }  false | { 2 }  { }  false | To check setP is NOT proper subset of SetQ |
| Change the values of setP and setQ  Print setP  Print setQ  setP < setQ | { }  { 2 }  true | { }  { 2 }  true | To check setP is proper subset of SetQ |

|  |  |  |  |
| --- | --- | --- | --- |
| Additional Tests for insert function | | | |
| Set set1, set2;  set1.insert(0);  set1.insert(-1);  set1.insert(-1);  set1.insert(99);  set1.insert(-15);  set1.insert(3);  Print set1 | { -15 -1 0 3 99 } | { -15 -1 0 3 99 } | Check negative values can be inserted in a correct way |
| Additional Tests for remove and size functions | | | |
| Print set1.size();  set1.remove(99);  print set1  Print set1.size(); | 5  { -15 -1 0 3 }  4 | 5  { -15 -1 0 3 }  4 | Check if the size changes in a correct way after removing a value |
| set2.size() | 0 | 0 | Check if size empty set returns 0 |
| set2.remove(4);  Print set2 | { } | { } | Check if removing a value from empty set does not change anything |
| Additional Tests for isEmpty functions | | | |
| set1.isEmpty() | false | false | Check if the non-empty Set returns “false” |
| set2.isEmpty() | true | true | Check if the empty Set returns “true” |
| Additional Tests for member functions | | | |
| set1.member(-15) | true | true | Check if the function returns “true” if the passed value is in the set |
| set2.member(0) | false | false | Check if the function returns “false” if set is empty |
| Additional Tests for union function | | | |
| Set setUnion12 = set1 + set2;  Print setUnion12 | { -15 -1 0 3 } | { -15 -1 0 3 } | Check if the function unions empty set and non-empty set |
| Additional Tests for intersection function | | | |
| Set setInter12 = set1 \* set2;  Print setInter12 | { } | { } | Check if the function returns empty if empty set and non-empty set were passed |