# Positive Testcases

## Initialize the Map

Note: Repeat with scenarios for different data types

interval\_map<unsigned int,char> myIntervalMap('A');

**Expect**

0,A

## Change the first element

Assign(1,2,B)

**Expect:**

0,A

1,B

## Assign a valid key/value range including the first element

Assign(1,3,C)

**Expect:**

0,A

1,C

2,C

## Append a valid key/value range to the map with no gap

Assign(3,6,D)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,D

## Overwrite entire existing key/value range with the same value

Assign(3,6,D)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,D

## Append a valid key/value range to the map with a gap

Assign(9,12,E)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,D

6,A

7,A

8,A

9,E

10,E

11,E

## Insert a valid key/value range into the middle of the map

Assign(5,8,F)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,F

6,F

7,F

8,A

9,E

10,E

11,E

## Insert a valid key/value range to overlap the end of the map

Assign(10,13,G)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,F

6,F

7,F

8,A

9,E

10,G

11,G

12,G

## Extend an existing key/value range with the same value

Assign(3,8,D)

**Expect:**

0,A

1,C

2,C

3,D

4,D

5,D

6,D

7,D

8,A

9,E

10,G

11,G

12,G

# Negative Testcases

## Initialize the Map

Note: Repeat with scenarios for different data types

interval\_map<unsigned int,char> myIntervalMap('A');

**Expect**

0,A

## Try to assign a keyBegin for the first key with a valid keyEnd

Assign(0,5,A)

**Expect:**

0,A

## Try to assign a keyBegin and keyEnd equal to first valid key

Assign(0,0,B)

**Expect:**

0,A

## Try to assign a keyBegin below the lowest valid key with keyEnd at first value

Assign(-1,1,B)

**Expect:**

0,A

## Try to assign a keyBegin below the lowest valid key with keyEnd valid

Assign(-1,3,B)

**Expect:**

0,A

## Try to assign a keyBegin and keyEnd below the lowest valid key

Assign(-2,-1,B)

**Expect:**

0,A

## Try to assign a keyBegin and keyEnd equal to first allowed key

Assign(1,1,B)

**Expect:**

0,A

## Try to assign a keyBegin and keyEnd below the lowest allowed key

Assign(-1,0,B)

**Expect:**

0,A

## Try to assign keyBegin within allowed key range with keyEnd after the max allowed

Assign(1, **std::numeric\_limits<unsigned int>::max()+1**,B)

**Expect:**

0,A

## Validate adjacent key values cannot be the same for lowest border

Assign(1,2,A)

**Expect:**

0,A

## Validate adjacent key values cannot be the same for adjacent border no gap

Assign(1,3,B)

**Expect:**

0,A

1,B

2,B

Assign(3,5,B)

**Expect:**

0,A

1,B

2,B

## Validate adjacent key values cannot be the same for adjacent border with gap

Assign(5,7,A)

**Expect:**

0,A

1,B

2,B

## Validate adjacent key values cannot be the same for lower gap border

Assign(1,2,A)

**Expect:**

0,A

1,B

2,B

## Validate adjacent key values cannot be the same for upper gap border

Assign(2,3,A)

**Expect:**

0,A

1,B

2,B

## Validate adjacent key values cannot be the same for upper embedded border before end

Assign(1,5,B)

**Expect:**

0,A

1,B

2,B

3,B

4,B

Assign(1,3,B)

**Expect:**

Error will be returned explicitly to show that the assignment did not take place, although theoretically no change will occur in the map in this case even if the assign takes place, that’s why it’s important that and error be displayed to show that the assign did not take place

0,A

1,B

2,B

3,B

4,B

## Validate adjacent key values cannot be the same for upper embedded border at end

Assign(1,4,B)

**Expect:**

Error will be returned explicitly to show that the assignment did not take place, although theoretically no change will occur in the map in this case even if the assign takes place, that’s why it’s important that and error be displayed to show that the assign did not take place

0,A

1,B

2,B

3,B

4,B