# Input Partitioning

List all the combinations of the equivalence classes for input parameters (arr and x):

| Partition | Equivalence Class for arr | Equivalence Class for x |
|---|---|---|
| P1 | Arr[] contains x | X is a valid integer |
| P2 | Arr[] does not contain x | X is a valid integer |
| P3 | Arr[] is empty | X is a valid integer |
| P4 | Arr[] is null | X is a valid integer |
| P5 | Arr[] contains more than 100 intergers | X is a valid integer |
| P6 | Arr[] not sorted in ascending order | X is a valid integer |
| P7 | Arr[] contains invalid integer element(s) | X is a valid integer |
| P8 | Arr[] contains x | X is not a valid integer |
| P9 | Arr[] does not contain x | X is not a valid integer |
| P10 | Arr[] is empty | X is not a valid integer |
| P11 | Arr[] is null | X is not a valid integer |
| P12 | Arr[] contains more than 100 intergers | X is not a valid integer |
| P13 | Arr[] not sorted in ascending order | X is not a valid integer |
| P14 | Arr[] contains invalid integer element(s) | X is not a valid integer |

List the test cases you designed to cover all the identified partitions. Test cases might be fewer than the partitions.

| Test | Partitions covered | arr input | x input | output |
|---|---|---|---|---|
| T1 | P1 | [10,20,30,40,50] | 50 | 4 |
| T2 | P2 | [10,20,30,40,50] | 1 | -1 |
| T3 | P3 | [ ] | 1 | -1 |
| T4 | P4 | none | 1 | Exception |
| T5 | P5 | [1,2,3…100,101] | 11 | Exception |
| T6 | P6 | [10, 9, 8,11,3] | 11 | Exception |
| T7 | P7 | [1.5,20,30,40,50] | 20 | Exception |
| T8 | P8, P14 | [1.5,20,30,40,50] | 1.5 | Exception |
| T9 | P9 | [10,20,30,40,50] | 1.5 | Exception |
| T10 | P10 | [ ] | 1.5 | Exception |
| T11 | P11 | none | 1.5 | Exception |
| T12 | P12 | [1,2,3…100,101] | 1.5 | Exception |
| T13 | P13 | [10, 9, 8,11,3] | 1.5 | Exception |

# Output Partitioning

List all the partitions, i.e., equivalence classes for the output domain.

| Partition | Equivalence Class for output |
|---|---|
| P1 | Returns a value of j that arr[j] == x |
| P2 | Return -1 |
| P3 | Exception |

List the test cases you designed to cover all the identified partitions. The test cases might be fewer than the partitions.

| Test | Partitions covered | arr input | x input | output |
|---|---|---|---|---|
| T1 | P1 | [10,20,30,40,50] | 50 | 4 |
| T2 | P2 | [10,20,30,40,50] | 1 | -1 |
| T3 | P3 | none | 1.5 | Exception |

# Boundary Value Robustness Testing

Test cases designed based on all combinations of boundary values for arr and x.

| Test | arr input | x input | output |
|---|---|---|---|
| T1 | [1] | 0 | -1 |
| T2 | [1] | 1 | 0 |
| T3 | [1,2,3...,99,100] | 1 | 0 |
| T4 | [1,2,3...,99,100] | 100 | 99 |
| T5 | [1,2,3...,99,100] | 0 | -1 |
| T6 | [1,2,3...,99,100] | 101 | -1 |
| T7 | [1,2,3...,99,100] | 1.5 | Exception |
| T8 | [ ] | 1 | -1 |
| T9 | none | 1 | Exception |

**Code execution:**

| No. | |
|---|---|
| | using System; |
| | class GFG |
| | { |
| 1 | static int binarySearch(int[] arr, int x) { |
| 2 | int low = 0, high = arr.Length − 1; |
| 3 | while (low <= high) { |
| 4 | int mid = low + (high − low) / 2; |
| 5 | if (arr[mid] == x) |
| 6 | return mid; |
| 7 | else if (arr[mid] < x) |
| 8 | low = mid + 1; |
| 9 | else |
| 10 | high = mid − 1; } |
| 11 | return −1; } |
| 12 | public static void Main() { |
| 13 | int[] arr = { 2, 3, 4, 10, 40 }; |
| 14 | int n = arr.Length; |
| 15 | int x = 10; |
| 16 | int result = binarySearch(arr, x); |
| 17 | if (result == −1) |
| 18 | Console.WriteLine("Element is not present in array"); |
| 19 | else |
| 20 | Console.WriteLine("Element is present at " + "index " + result); } |
| | } |

# Statement Coverage

Please indicate which language version of source code you are using: C#

List the test cases designed following the Statement Coverage method.

| Test | Statement (line#) covered | arr input | x input | output |
|---|---|---|---|---|
| T1 | 1, 2, 3, 4, 5, 6, 12 - 16, 19, 20 | [2,3,4,10,40] | 4 | "Element is present at index 2" |
| T2 | 1, 2, 3, 4, 7, 8, 12 - 16, 19, 20 | [2,3,4,10,40] | 10 | "Element is present at index 3" |
| T3 | 1, 2, 3, 4, 9, 10, 12 - 16, 19, 20 | [2,3,4,10,40] | 2 | "Element is present at index 0" |
| T4 | 1, 2, 11, 12 – 16, 17, 18 | [ ] | 1 | "Element is not present in array" |
| T5 | 1, 2, 3, 4, 9, 10, 11, 12 - 16, 17, 18 | [2,3,4,10,40] | 1 | "Element is not present in array" |
| T6 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12 -16, 19, 20 | [2,3,4,10,40] | 3 | "Element is present at index 1" |

# Decision Coverage

Please indicate which language version of source code you are using: C#

Identify all sides of every decision point, and the inputs that result in each decision side:

| ID | Decision | arr input | x input |
|---|---|---|---|
| D1 | While (low <= high) true | [1, 2, 3] | 2 |
| D2 | While (low <= high) false | [ ] | 2 |
| D3 | If (arr[mid] == x) true | [1, 2, 3] | 2 |
| D4 | If (arr[mid] == x) false | [1, 2, 3] | 3 |
| D5 | If (arr[mid] < x) true | [1, 2, 3] | 3 |
| D6 | If (arr[mid] < x) false | [ 1, 2, 3] | 2 |
| D7 | If (result == -1) true | [ ] | 2 |
| D8 | If (result == -1) false | [1, 2, 3] | 2 |

List the tests you designed to cover the decisions in the above table. Test cases might be fewer than the decisions.

| Test | Decisions covered | arr input | x input | output |
|---|---|---|---|---|
| T1 | D1, D3, D6, D8 | [1, 2, 3] | 2 | "Element is present at index 1” |
| T2 | D2, D7 | [ ] | 2 | "Element is not present in array" |
| T3 | D4, D5 | [1, 2, 3] | 3 | "Element is present at index 2” |

# Loop Coverage

Please indicate which language version of source code you are using: C#

Identify different cases for the loop structure, design inputs to result in each case, and finally, figure out the expected output for each input.

| Test | Loop body | arr input | x input | Output |
|---|---|---|---|---|
| T1 | Zero times | [ ] | 2 | "Element is not present in array" |
| T2 | Once | [1] | 1 | "Element is present at index 0” |
| T3 | Twice | [1, 2] | 2 | "Element is present at index 1” |
| T4 | Many times | [2, 3, 4, 5, 6] | 6 | "Element is present at index 4” |

# Path Coverage

Please indicate which language version of source code you are using: C#

Identify all possible paths:

| Path | Line# in the path |
|---|---|
| PT1 | 1, 2, 3, 11, 12 - 16, 17, 18, EXIT |
| PT2 | 1, 2, 3, 11, 12 - 16, 17, 19, 20, EXIT *(invalid)* |
| PT3 | 1, 2, 3, 4, 5, 6, 12 - 16, 17, 18, EXIT *(invalid)* |
| PT4 | 1, 2, 3, 4, 5, 6, 12 - 16, 17, 19, 20, EXIT |
| PT5 | 1, 2, 3, 4, 5, 7, 8, 12 - 16, 17, 18, EXIT |
| PT6 | 1, 2, 3, 4, 5, 7, 8, 12 - 16, 17, 19, 20, EXIT |
| PT7 | 1, 2, 3, 4, 5, 7, 9, 10, 12 - 16, 17, 18, EXIT |
| PT8 | 1, 2, 3, 4, 5, 7, 9, 10, 12 - 16, 17, 19, 20, EXIT |
| PT9 | 1, 2, 3, 4, 5, 7, 8, 3, 4, 5, 6, 12 - 16, 17, 18, EXIT *(invalid)* |
| PT10 | 1, 2, 3, 4, 5, 7, 8, 3, 4, 5, 6, 12 - 16, 17, 19, 20, EXIT |
| PT11 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 6, 12 - 16, 17, 18, EXIT *(invalid)* |
| PT12 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 6, 12 - 16, 17, 19, 20, EXIT |
| PT13 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 7, 8, 12 - 16, 17, 18, EXIT *(invalid)* |
| PT14 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 7, 8, 12 - 16, 17, 19, 20, EXIT |
| PT15 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 7, 8, 3, 4, 5, 6, 12 - 16, 17, 18, EXIT *(invalid)* |
| PT16 | 1, 2, 3, 4, 5, 7, 9, 10, 3, 4, 5, 7, 8, 3, 4, 5, 6, 12 - 16, 17, 19, 20, EXIT |

List the test cases you designed to cover all the identified paths:

| Test | Paths covered by test | arr input | x input | Output |
|---|---|---|---|---|
| T1 | PT1 | [ ] | 1 | "Element is not present in array" |
| T2 | PT4 | [1,2] | 1 | "Element is present at index 0” |
| T3 | PT5 | [1,2,3] | 4 | "Element is not present in array" |
| T4 | PT6 | [1,2,3] | 3 | "Element is present at index 2” |
| T5 | PT7 | [1,2,3] | 0 | "Element is not present in array" |
| T6 | PT8 | [1,2,3] | 1 | "Element is present at index 0” |
| T7 | PT10 | [1,2] | 2 | "Element is present at index 1” |
| T8 | PT12 | [1,2,3,4,5] | 1 | "Element is present at index 0” |
| T9 | PT14 | [1,2,3,4,5,6] | 2 | "Element is present at index 1” |
| T10 | PT16 | [1,2,3,4,5] | 2 | "Element is present at index 1” |