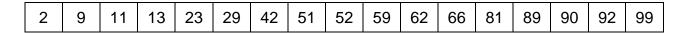
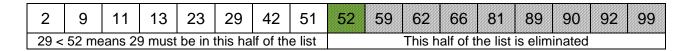
## **Searching Algorithm – Binary Search**

Binary search is a more efficient algorithm compared to sequential search, but it requires the list to be sorted in order. Binary search compares the target value to the item in the middle of the list. If they are not equal, based on how the target value is relative to the middle item, the algorithm determines the half of the list that the target lies on and eliminates the other half. The remaining list continues to be searched until successful.

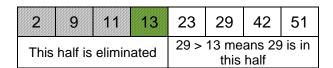
Supposed we are searching for 29 in the following ascendingly sorted list.



Initially, 29 is compared to the middle item, 52, which is not equal. Since 29 is smaller than the middle item, if it exists on the list, it must be in the first half of the list. Therefore, the algorithm eliminates the second half of the list and continues to search on the first half.



The process repeats with the remaining list. 29 is now compared to the middle item of the new list, 13, which is not equal. Since 29 is greater than 13, it must be in the second half of the list if it exists. Thus, the first half of the list is eliminated.



In the remaining list, 29 is the middle item, which is the target item. The search is successful!



## **Implementation**

To implement binary search, two variables top and bottom are used to keep track of the upper and lower bound of the list to be searched. These two variables are updated as half of the list is eliminated based on the comparison of the target and middle item.

```
binarySearch(list L, item x)
  set bottom to 0
  set top to length of list - 1
  set found to false
  set i to -1

while (bottom is less than or equal to top and found is false)
  set middle to (bottom + top) / 2
  if (x matches item at index middle)
     set found to true
     set i to middle
  else if (x > item at index middle) //not in the bottom half
     set bottom to middle + 1
  else //not in the top half
     set top to middle -1

return i
```