Binary Search

Examples:

1. When to use Binary Search
2. Trace the steps - found
3. Trace the steps – not found
4. Binary Search – Function Definition

5*. Find a range (target not unique)

1. What condition must be true before a binary search can be used on a list?

1. What condition must be true before a binary search can be used on a list?

The list must be sorted!

1. What condition must be true before a binary search can be used on a list?

The list must be sorted!

Binary search should be used to search large (sorted) lists.

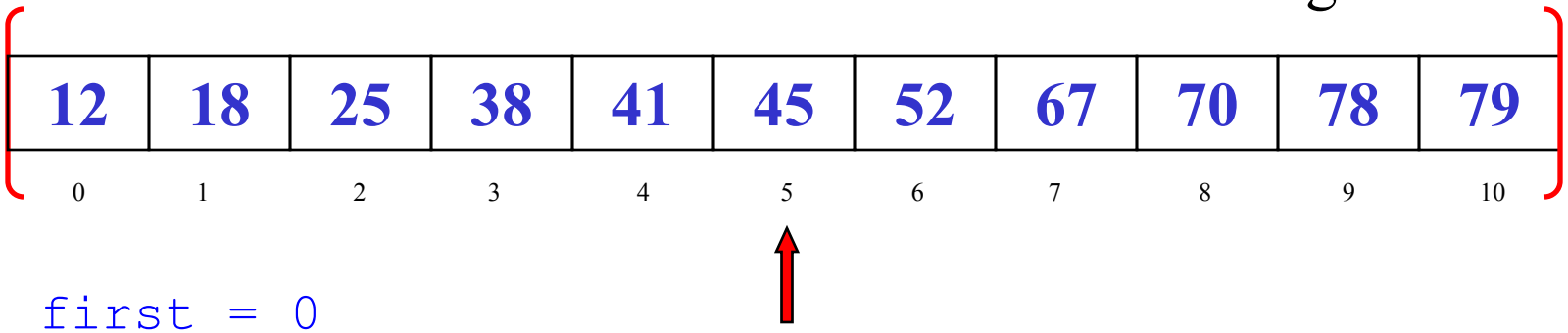
2. Trace the steps of the binary search algorithm to search the following list:

target = 38

12	18	25	38	41	45	52	67	70	78	79
0	1	2	3	4	5	6	7	8	9	10

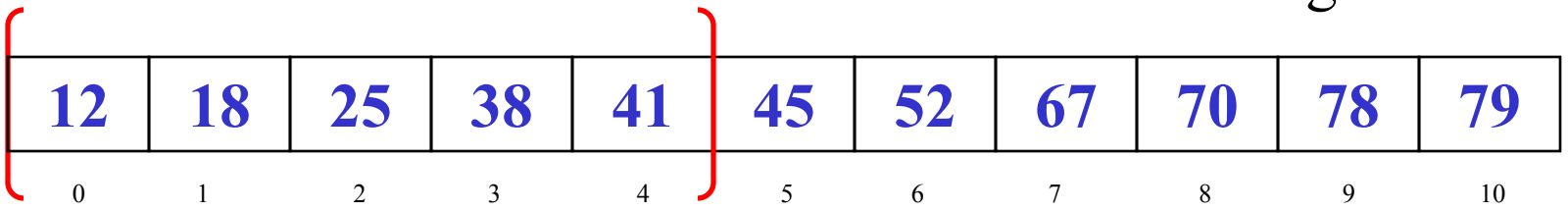
target = 38

target = 38



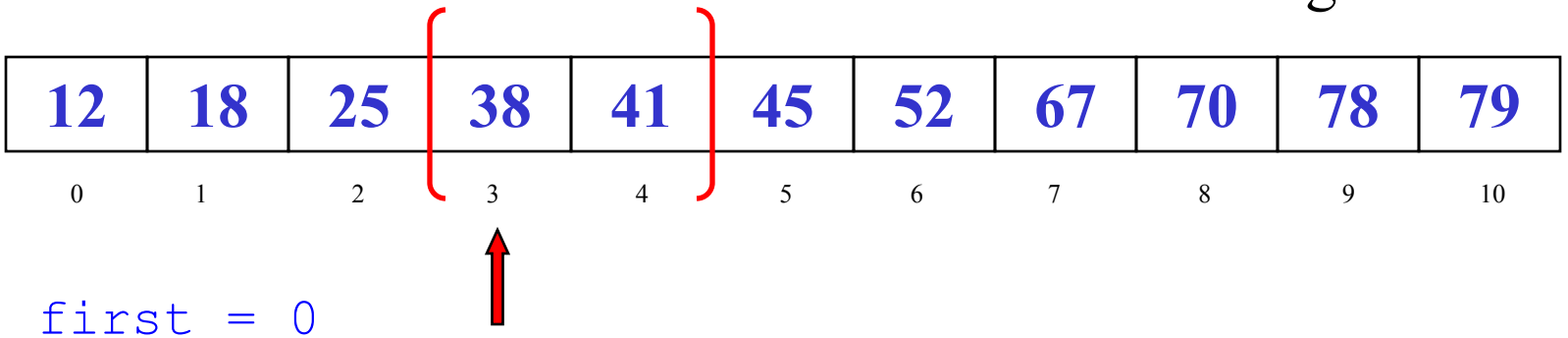

```
first = 0
last  = 10
mid = (first + last)/2 = 5
38 < list[5] => search the lower half:
      last = mid - 1 = 4
```

target = 38




```
first = 0
last  = 10
mid = (first + last)/2 = 5
38 < list[5] => search the lower half:
      last = mid - 1 = 4

mid = (0 + 4)/2 = 2
38 > list[2] => search the upper half;
      first = mid + 1 = 3
```

target = **38**



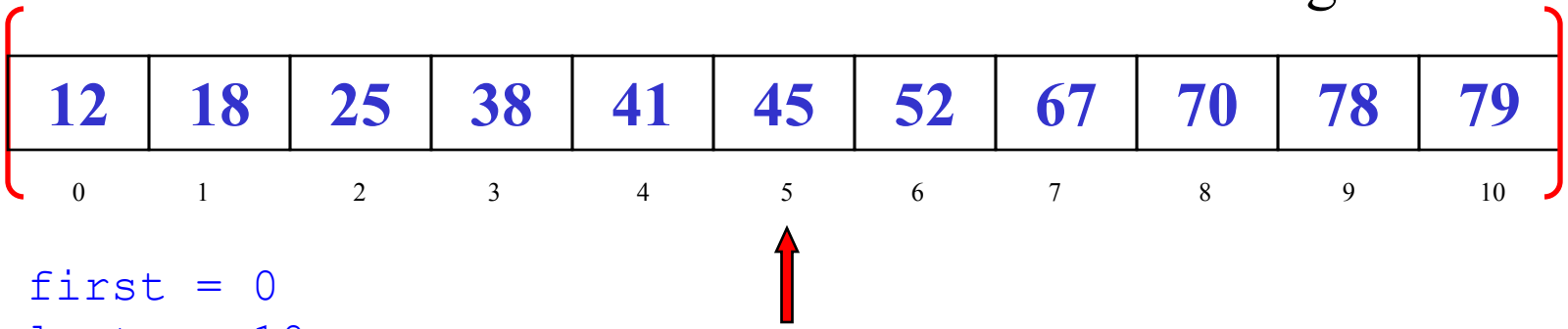

```
first = 0
last  = 10
mid = (first + last)/2 = 5
38 < list[5] => search the lower half:
      last = mid - 1 = 4

mid = (0 + 4)/2 = 2
38 > list[2] => search the upper half;
      first = mid + 1 = 3

mid = (3 + 4)/2 = 3
38 == list[3] STOP: target found at index 3!
```

3. Trace the steps of the binary search algorithm to search the following list:

target = 60

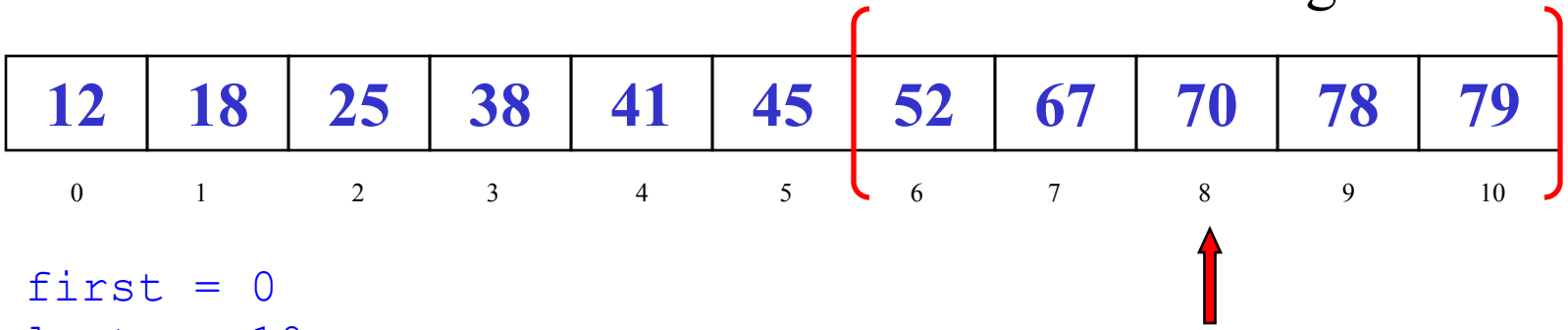
12	18	25	38	41	45	52	67	70	78	79
0	1	2	3	4	5	6	7	8	9	10

target = 60



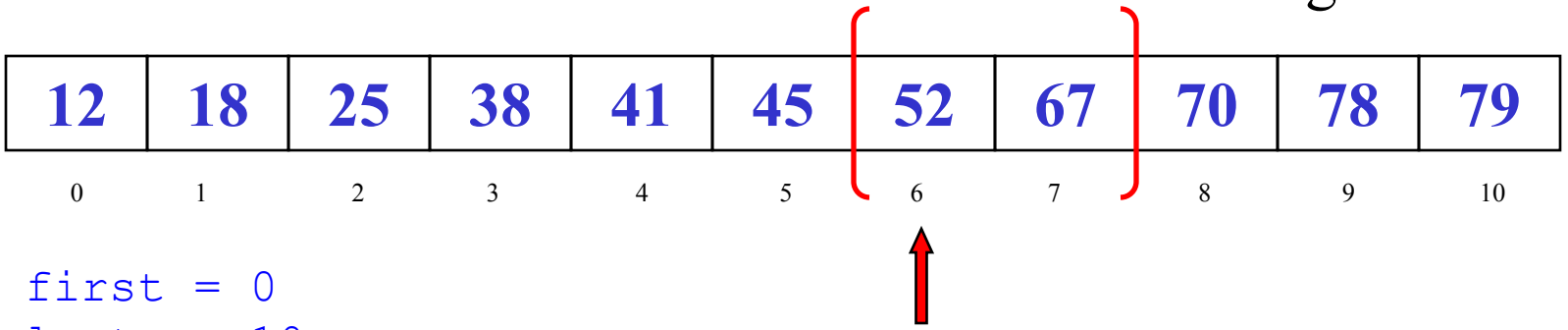

```
first = 0
last  = 10
mid = (first + last)/2 = 5
60 > list[5] => search the upper half
       first = mid + 1 = 6
```

target = 60



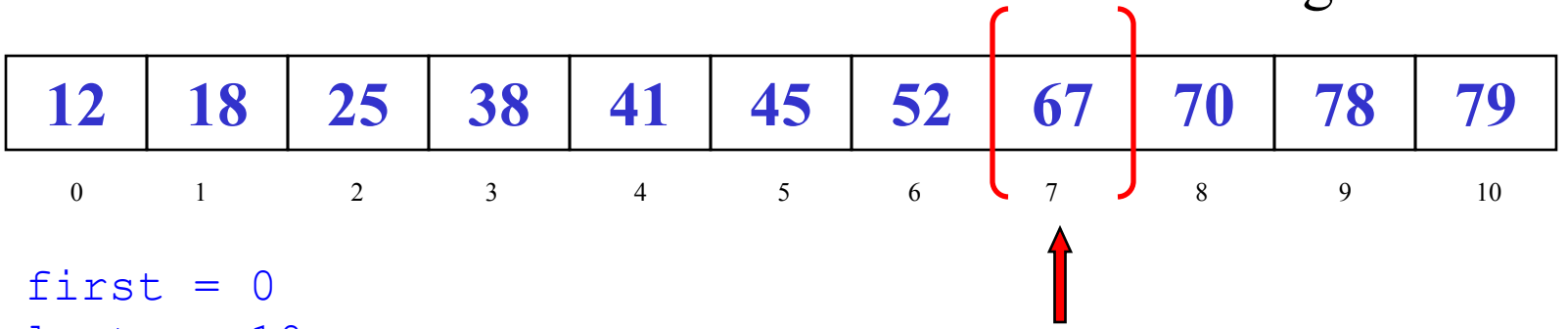

```
first = 0
last  = 10
mid = (first + last)/2 = 5
60 > list[5] => search the upper half:
       first = mid + 1 = 6
mid = (6 + 10)/2 = 8
60 < list[8] => search the lower half:
      last = mid - 1 = 7
```

target = 60




```
first = 0
last  = 10
mid = (first + last)/2 = 5
60 > list[5] => search the upper half:
       first = mid + 1 = 6
mid = (6 + 10)/2 = 8
60 < list[8] => search the lower half:
      last = mid - 1 = 7
mid = (6 + 7)/2 = 6
60 > list[6] => search the upper half
       first = mid + 1 = 7
```

target = 60




```
first = 0
last  = 10
mid = (first + last)/2 = 5
60 > list[5] => search the upper half:
       first = mid + 1 = 6
mid = (6 + 10)/2 = 8
60 < list[8] => search the lower half:
      last = mid - 1 = 7
mid = (6 + 7)/2 = 6
60 > list[6] => search the upper half
       first = mid + 1 = 7
mid = (7 + 7)/2 = 7
60 < list[7] => search the lower half
      last = mid - 1 = 6
first = 7 > 6 = last => STOP: target not found!
```

4. Binary Search – Function Definition

```
int binarySearch(const int array[], int size, int value)
{
    int first = 0,          // First array element
        last = size - 1,    // First array element
        middle,             // Mid point of search
        position = -1;      // Position of search value

    while (position == -1 && first <= last)
    {
      middle = (first + last) / 2;     // Calculate mid point
      if (array[middle] == value)      // If value is found at mid
          position = middle;
      else if (array[middle] > value) // If value is in lower half
         last = middle - 1;
      else
         first = middle + 1;           // If value is in upper half
   }
   return position;
}
```

5*. Show how a sorted array of unique numbers can be searched for a certain number using binary search. Code a calling statement for binary search and print appropriate messages.

```
int binarySearch(double array[], int size, float value);

int main( void )
{
   double list[100] = {2, 3, 5, 7, 8, 9, 10, 20, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;



   return 0;
}
```

```
int binarySearch(double array[], int size, double value);

int main( void )
{
   double list[100] = {2, 3, 5, 7, 8, 9, 10, 20, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;









   return 0;
}
```

```
int binarySearch(double array[], int size, double value);

int main( void )
{
   double list[100] = {2, 3, 5, 7, 8, 9, 10, 20, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;

   index = binarySearch( list, length, target);




   return 0;
}
```

```
int binarySearch(double array[], int size, double value);

int main( void )
{
   double list[100] = {2, 3, 5, 7, 8, 9, 10, 20, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;

   index = binarySearch( list, length, target);
   if ( index != -1 )
      cout << target << " found at index " << index;
   else
      cout << target << " not found!";
   cout << endl;
   return 0;
}
```

5*. Show how a sorted array of numbers can be searched for a certain number using binary search. Code a calling statement for binary search and print appropriate messages.

```
int binarySearch(double array[], int size, double value);

int main( void )
{
   double list[100] = {5, 5, 8, 8, 8, 8, 8, 8, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;



   return 0;
}
```

```
int binarySearch(double array[], int size, double value);

int main( void )
{
   double list[100] = {5, 5, 8, 8, 8, 8, 8, 8, 50, 90};
   int    length    = 10;
   double target    = 8;
   int    index;

   index = binarySearch( list, length, target);
   if ( index != -1 )
      // here we should print a range: from index 2 to 7, inclusive
   else
      cout << target << " not found.\n";

   return 0;
}
```

```
double list[100] = {5, 5, 8, 8, 8, 8, 8, 8, 50, 90};


// here we should print a range: from index 2 to 7, inclusive!
   z = index + 1;
   while( z < length && list[z] == list[index] )
       z++;
   z--;
```

```
double list[100] = {5, 5, 8, 8, 8, 8, 8, 8, 50, 90};


// here we should print a range: from index 2 to 7, inclusive!
   z = index + 1;
   while( z < length && list[z] == list[index] )
       z++;
   z--;

   a = index - 1;
   while( a >= 0 && list[a] == list[index] )
       a--;
   a++;

   cout << target << " found from index "
        << a << " to " << z << endl;
```

Binary Search

Examples:

- ✓ 1. When to use Binary Search
- ✓ 2. Trace the steps - found
- ✓ 3. Trace the steps – not found
- ✓ 4. Binary Search – Function Definition
- ✓ 5*. Find a range (target not unique)