PROGRAMMING FOR PROBLEM SOLVING -- PPS

SUBJECT CODE- BTPS-101-18

By
Dr.A.Deepa,AP,Chandigarh Engineering College

PPS SYLLABUS

Unit 4

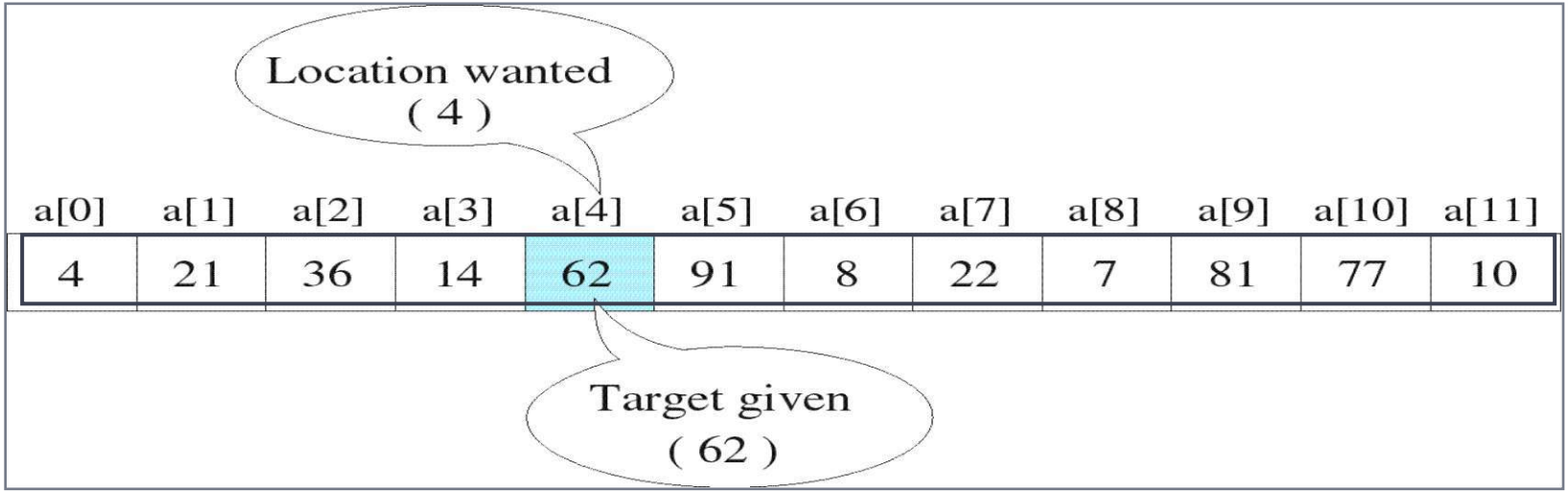
Basic Algorithms (6 lectures)

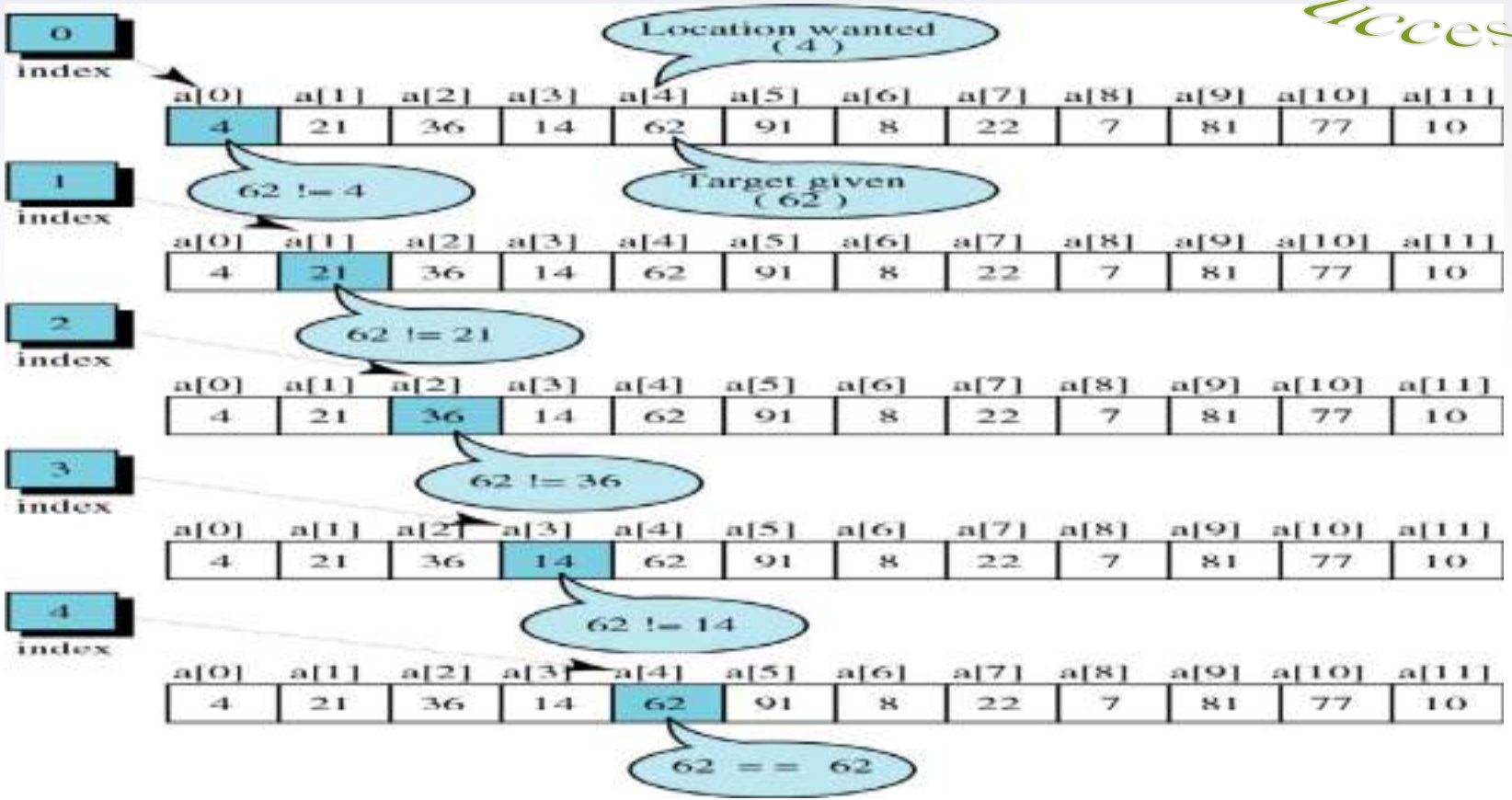
Searching, Basic Sorting Algorithms (Bubble, Insertion and Selection), Finding roots of equations, notion of order of complexity through example programs (no formal definition required)

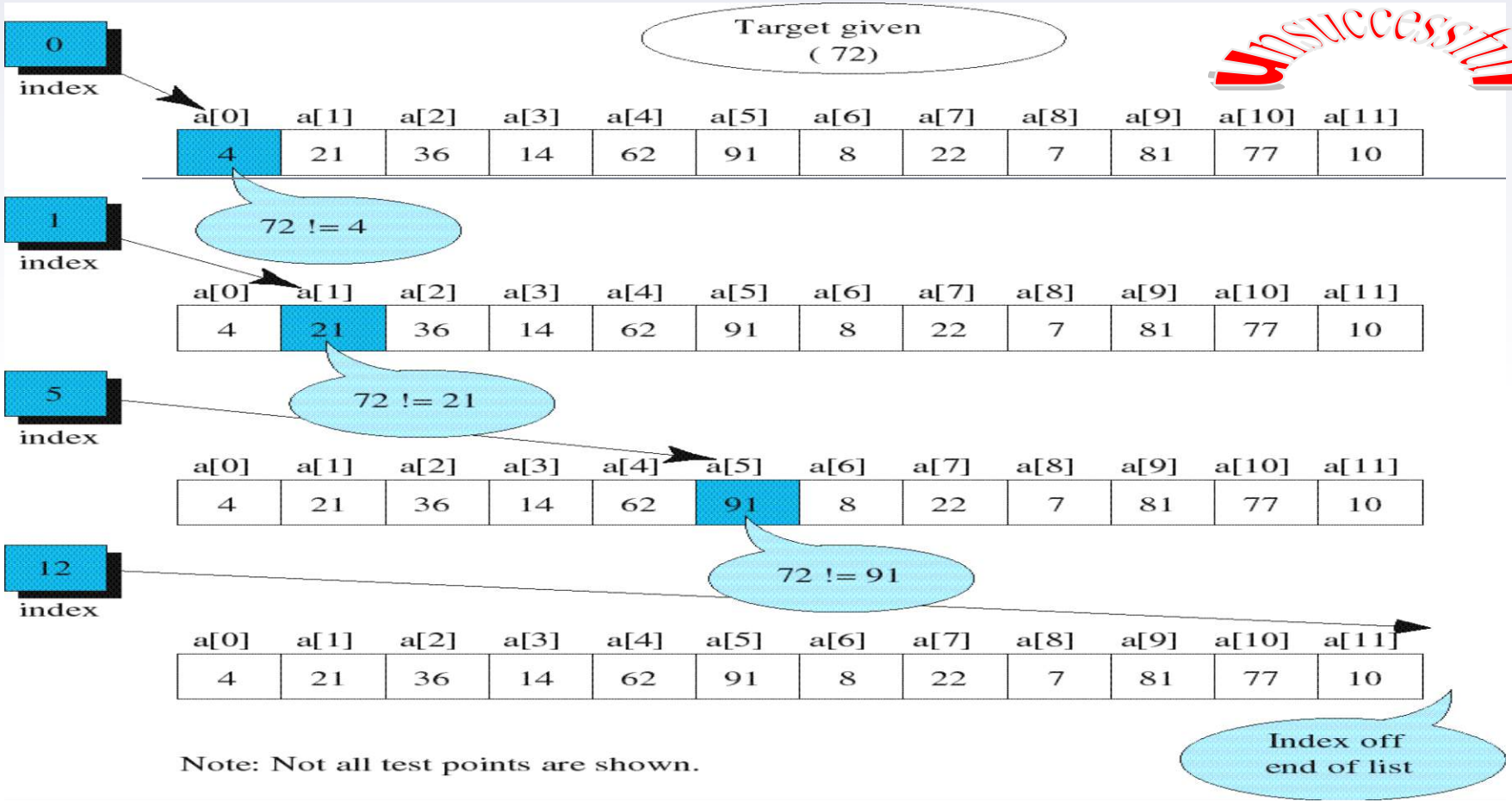
By Dr.A.Deepa,AP,Chandigarh Engineering College

Searching

- The process used to find the location of a target among a list of objects
- Searching an array finds the index of first element in an array containing that value




0
index
Location wanted
(4)
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
1
index
62 != 4
Target given
(62)
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
2
index
62 != 21
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
3
index
62 != 36
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
4
index
62 != 14
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
62 == 62


Target given
(72)
Unsuccessful
0
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
72 != 4
1
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
72 != 21
5
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
72 != 91
12
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
Index off
end of list
Note: Not all test points are shown.

Unordered Linear Search

- Search an unordered array of integers for a value and return its index if the value is found. Otherwise, return -1.
- A[0] A[1] A[2] A[3] A[4] A[5] A[6] A[7]

14	2	10	5	1	3	17	2

- **Algorithm:**

Start with the first array element (index 0)

while(more elements in array){

if value found at current index, **return index**;

Try next element (increment index);

}Value not found, return -1;

Unordered Linear Search

```
// Searches an unordered array of integers
int search(int data[],  //input: array
          int size,    //input: array size
           int value){  //input: search value
    // output: if found, return index;
    //         otherwise, return –1.
    for(int index = 0; index < size; index++){
        if(data[index] == value)
            return index;
        }    return -1;}
```

Linear Search Algorithm

- LINEAR_SEARCH(A, N, VAL)
- Step 1: [INITIALIZE] SET POS = -1 Step 2: [INITIALIZE] SET I = 1
- Step 3: Repeat Step 4 while I<=N
- Step 4: IF A[I] = VAL
- SET POS = I
- PRINT POS
- Go to Step 6
- [END OF IF]
- SET I = I + 1
- [END OF LOOP]
- Step 5: IF POS = –1
- PRINT VALUE IS NOT PRESENT IN THE ARRAY
- [END OF IF]
- Step 6: EXIT

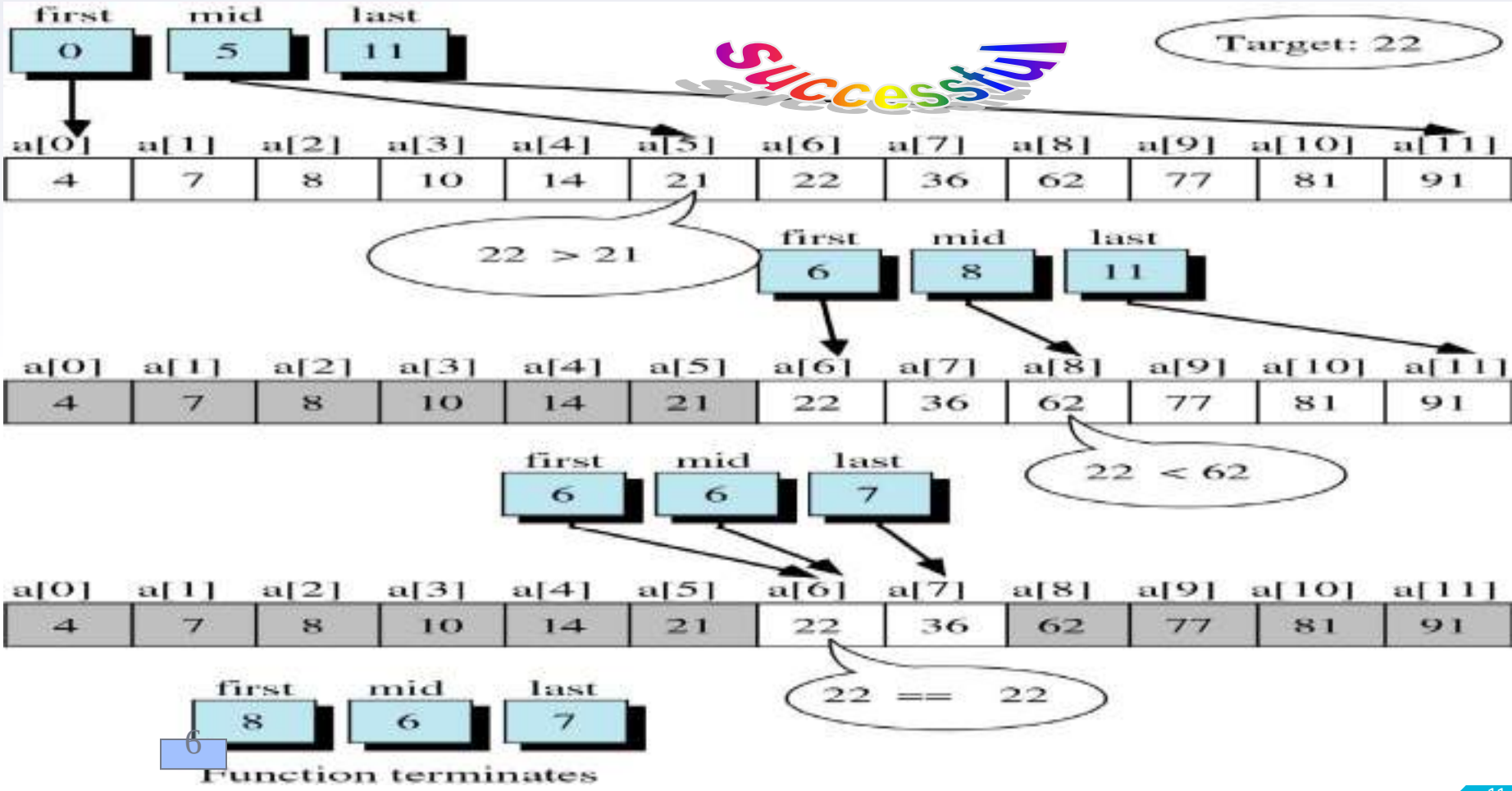
Program to perform Linear Search

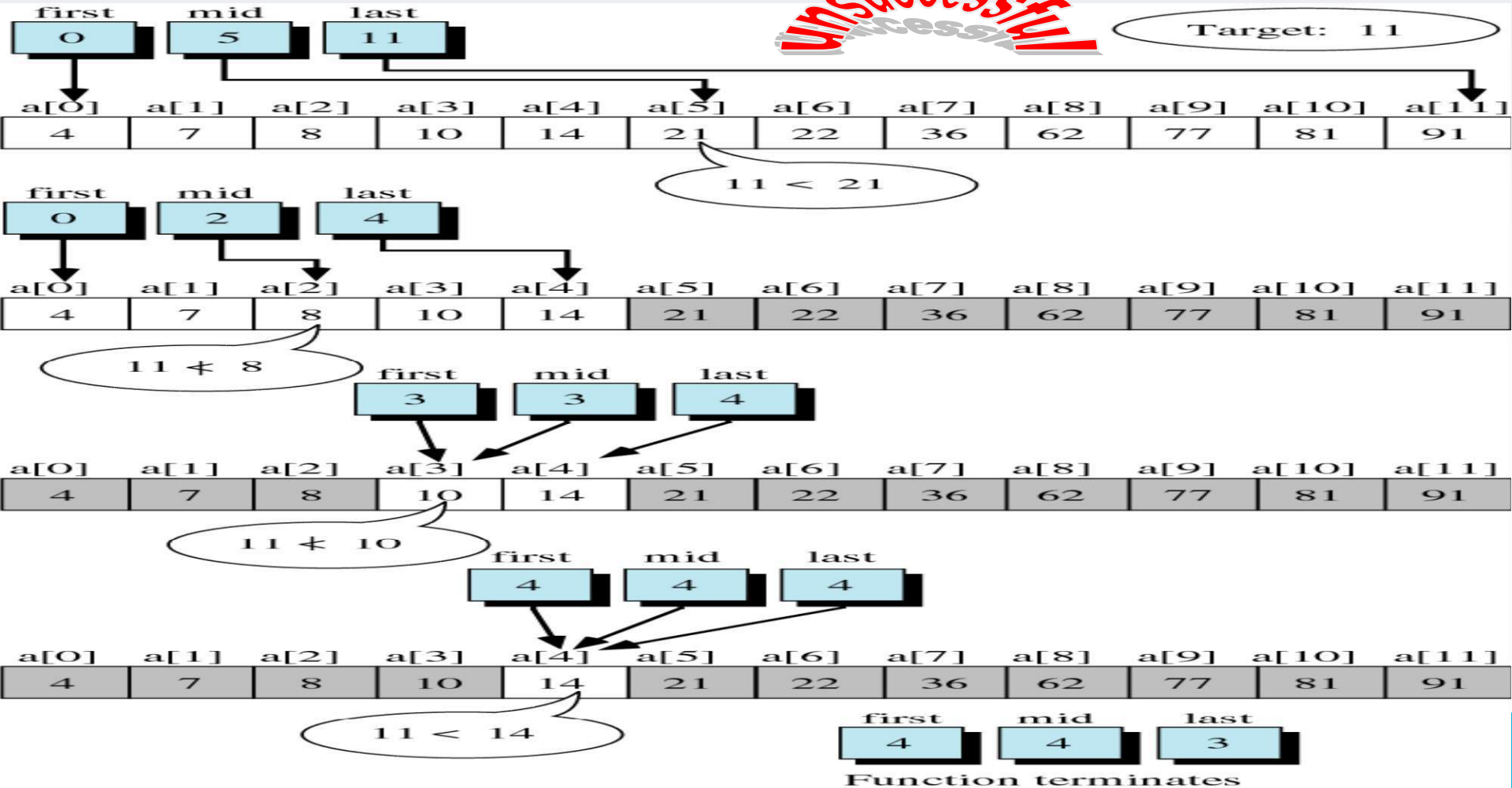
```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[11],b,c;
 clrscr();
 printf("Enter an array  ");
 for(b=0;b<11;b++)
 {
   scanf("%d",&a[b]);
 }
 printf("\nWhich value you want to search");
 scanf("%d",&c);
 for(b=0;b<11;b++)
 {
 if(a[b]==c)
  {
   printf("\nLocation of that value is %d",b+1);
   break;
  }
 }
 if(b==11)
 printf("\nValue not found in array");
 getch();
}
```

Binary Search

- Search an ordered array of integers for a value and return its index if the value is found. Otherwise, return -1.

- Binary search skips over parts of the array if the search value cannot possibly be there.


first
mid
last
0
5
11
Target: 22
Successful
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
22 > 21
first
mid
last
6
8
11
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
22 < 62
first
mid
last
6
6
7
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
22 == 22
first
mid
last
8
6
7
6
Function terminates


Unsuccessful
Target: 11
first
mid
last
0
5
11
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
11 < 21
first
mid
last
0
2
4
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
11 ≠ 8
first
mid
last
3
3
4
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
11 ≠ 10
first
mid
last
4
4
4
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
11 < 14
first
mid
last
4
4
3
Function terminates

Binary Search Algorithm

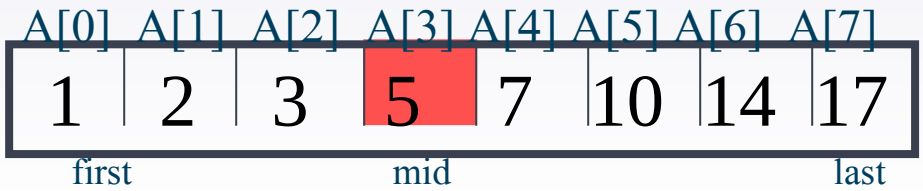
- Step 1: [INITIALIZE] SET BEG = lower_bound
- END = upper_bound, POS = - 1
- Step 2: Repeat Steps 3 and 4 while BEG <= END
- Step 3: SET MID = (BEG + END)/2
- Step 4: IF A[MID] = VAL SET POS = MID PRINT POS Go to Step 6
- ELSE IF A[MID] > VAL
- SET END = MID - 1
- ELSE
- SET BEG = MID + 1
- [END OF IF]
- [END OF LOOP]
- Step 5: IF POS = -1
- PRINT "VALUE IS NOT PRESENT IN THE ARRAY"
- [END OF IF] Step 6: EXIT

Binary Search

- Binary search is based on the “divide-and-conquer” strategy which works as follows:
  - Start by looking at the middle element of the array
    - 1. If the value it holds is lower than the search element, eliminate the first half of the array from further consideration.
    - 2. If the value it holds is higher than the search element, eliminate the second half of the array from further consideration.
  - Repeat this process until the element is found, or until the entire array has been eliminated.

Example: binary search

- 14 ?

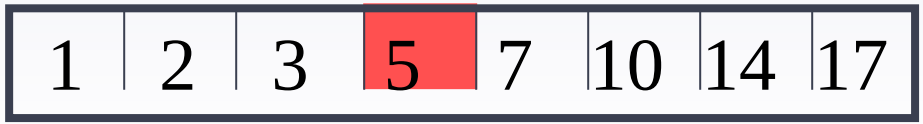


In this case,
(data[middle] == value)
return middle;

Example: binary search

A[0] A[1] A[2] A[3] A[4] A[5] A[6] A[7]

- 8 ?



first mid last

A[4] A[5] A[6] A[7]

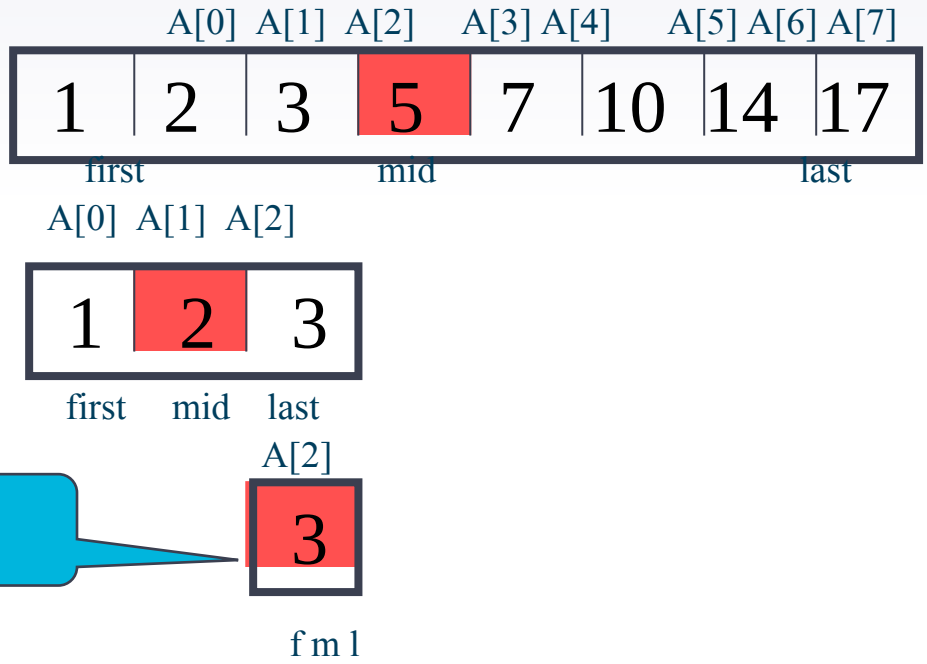
first mid last

```
In this case, (first == last)
    return -1;
```

A[4] f m l

Example: binary search

- 4 ?




```
In this case, (first == last)
    return -1;
```

Thank You

Queries????