• The loop in lines 4–11 continue upto $i=n-2$. ∴ Finally, we get $list[0] \le list[1] \le \ldots \le list[n-2]$.

• We can easily show that $list[n-2] \le list[n-1]$.

• ∴ Function sort(list, n) is correct □

Ex: (Binary Search) Assume that we have $n \geq 1$ distinct integers that are already sorted and stored in the array list. $list[0] \le list[1] \le \ldots \le list[n-1]$

Q) Is integer searchnum present in list? If yes, return index i s.t. $list[i] =$ searchnum. If No, return -1.

Let left and right, respectively, denote the left and right ends of the list to be searched. Initially, left $= 0$ and right $= n-1$. Let middle $=$ (left $+$ right)$/2$ be the middle position in the list.

(1) searchnum $<$ list[middle]. If searchnum is present, it must be in between the posns. 0 to middle-1. ∴ right $=$ middle-1

(2) searchnum $=$ list[middle] return middle

(3) searchnum $>$ list[middle]. If searchnum is present, it must be in between the posns. middle$+1$ and $n-1$. ∴ left $=$ middle$+1$

• If searchnum has not been found and there are still integers to check, we recalculate the middle and continue the search.

First attempt

```
while(there are more integers to check){
   middle=(left+right)/2;
   if(searchnum<list[middle])
      right=middle-1;
   else if(searchnum==list[middle])
      return middle;
   else left=middle+1; }
```

Two subtasks:

1) determining if there are any integers left to check

2) comparing searchnum with list[middle]