7.46) (b)

```
public static int[] sort(int []arr, int n)
{
// Define and initialize an array containing 4 elements
int []arr = new int[]{a1,a2,a3,a4};
// Compare the first two elements
if (a[0] < a[1]) {
// Swap these elements
a1 = a[0];
a2 = a[1];
}
else {
// If the other element is smaller, swap with it
a1 = a[1];
a2 = a[0];
}
// Compare the last two elements among themselves
if (a[2] < a[3]) {
a3 = a[2];
a4 = a[3];
}
Else {
a3 = a[3];
a4 = a[2];
}
// Compare the smaller elements obtained from the above comparisons
if (a1 < a3) {
a11 = a1;
mid1 = a3;
else {
// Create the new value of a3 after comparison
a11 = a3;
mid1 = a1;
}
// The second and the last elements are compared
if (a2 > a4) {
highest = a2;
mid2 = a4;
}
else {
highest = mid2;
mid2 = mid1;
}
// Compare the new values
if (mid < mid2) {
return (a11,mid1,mid2,highest);
```

```
}
else {
// Return the sorted order
return (a11,mid2,mid1,highest);
}
}
```

7.53)

(a)

```
// Method to check wheter the elements present
// or not whose sum is given
boolean sumOfNum(int ArrayList[])
{
// Accept the variable K,
// n is the size of the array
int K,n,sum;
// Start the loop to find the sum of each two elemnts in the loop
for (int count=1; count …)
{
// Start another loop
for (int num=1; num …)
{
// Value of sum changes at each step
sum = ArrayList[count]+ArrayList[num];
if (sum==K)
// Returns true
return true;
}
}
return false;
}
```

(b)

```
boolean sumOfNum(int ArrayList[])
{
int K,sum,n;
int count = 0;
// Apply Quicksort to sort the array
Quicksort(ArrayList);
// Start the while loop from 1 to n
while (count<n)
{
// Calculates sum of Array elements
// at the count position and at n position
sum = ArrayList[count] + ArrayList[n];
```

```
// Check whether the sum is equal
// to K or not
if (sum == K)
return true;
else if (sum < K)
count++
else
n--;
}
return false;
}
```