Question 1

Solution 1

```
for (i=0 to n)
  {
   if (arr[i] == value)
  }
```

Solution 2

```
void insertion_sort (int arr[], int n)
  {
   for (int i=1; i<n; i++)
    {
      j=i-1;
     x= arr[i];
    while (j>-1 && arr[j]>x)
    {
      arr[j+1] = arr[j];
          j--;
    }
    arr[j+1]=x;
   }
  }
```

Recursion

```
void insertion_sort (int arr[], int n)
 {
   if (n<=1)
    return;
 insertion_sort (arr, n-1);
 int last = arr[n-1];
 int j=n-2;
 while (j>=0 && arr[j]>last)
    {
      arr[j+1] = arr[j];
```

arr[j+1] = last

}

Insertion sort is called 'online sort' because it does not need to known anything about what values it will sort and information is required while algorithm is running.

Other sorting Algorithm

1. Bubble sort
2. Quick sort
3. Merge sort
4. Selection sort
5. Heap sort

Question 3

Sort Algo	Best	Worst	Average
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble sort	$\Omega(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$\Omega(n)$	$O(n^2)$	$O(n^2)$
Heap sort	$\Theta(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick sort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$
Merge sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$

Question 4

INPLACE SORTING	STABLE	ONLINE SORTING
Bubble sort	Merge sort	Insertion sort
Selection sort	Bubble sort	
Insertion sort	Insertion sort	
Quick sort	Count sort	
Heap sort		

Solutions

```
int bsearch (int arr[], int l, int r, int key)
{
  while(l<=r)
  {
    int m = ((l+r)/2);
    if (arr[m] == key)
        return m;
    Else if (key < arr[m])
        r = m-1;
    Else
        l = m+1;
  }
  return -1;
}
```

Recursive

```
int bsearch (int arr[], int l, int r, int key)
  {
    while (l<=r) {
    int m = (l+r)/2);
    if (key == arr[m])
        return m;
    Else if (key < arr[m])
      return bsearch (arr, l, mid-1, key);
    Else
      return bsearch (arr, mid+1, r, key);
    }
    return -1;
  }
```

time complexity

linear search - O(n)

Binary search - O(log n)

(6)

$T(n) = T(n/2) + 1$ —①

$T(n/2) = T(n/4) + 1$ —②

$T(n/4) = T(n/8) + 1$ —③

$T(n) = T(n/2) + 1$

$= T(n/4) + 1 + 1$

$= T(n/8) + 1 + 1 + 1$

$= T(n/2^k) + 1$ (K times)

let $g^k = n$

$K = \log n$

$T(n) = T(n/n) + \log n$

$T(n) = T(1) + \log n$

$T(n) = O(\log n)$ → Answer

ques-7

solution 7

```
for (i=0; i<n; i++)
  {
    for (int j=0; j<n; j++)
      {
        if (a[i]+a[j] == k)
          cout<<i<<j;
      }
  }
```

ques 8

Quicksort is fastest general purpose sort. In most practical situations quicksort is the method of choice as stability is important and space is available might be best

ques-9

Solution 9

A pair (A[i], A[j]) is said to be inversion if

∵ A[i] > A[j]

∴ i < j

Total no. of inversion in given array are 31 using merge sort.

question 10

Solution 10

Worst case $O(n^2)$:→ the worst case occurs when the pivot element is an extreme (smallest / largest) element. this happens when input array is sorted or reverse sorted. and either first or last element is selected as pivot.

Best case $O(n \log n)$:- the best case occurs when we will select as pivot element as a mean element.

question 11

Solution 11

Merge sort:

Best case : $T(n) = 2T(n/2) + O(n)$
worst case : $T(n) = 2T(n/2) + O(n)$ } $O(n \log n)$

Quick Sort

Best case: $T(n) = 2T(n/2) + O(n) \rightarrow O(n \log n)$
worst case: $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

ques 12

question

Answer 12:

```
for (int i=0; i<n-1; i++)
{
    int min = i;
for (int j=i+1; j<n; j++)
```

3.6

```
}if (a[min]>a[j])
        min=j;
}
int key= a[min];
while (min>i)
{
  a[min] =a[min-j];
    min--;
}
a[i]=key;
}
```

ques13

Solution A better version of bubble sort, known as modified bubble sort, including a flag that is set if a exchange is made after an entire pass over. If no exchange is made then it should be called the array is already order because no two elements need to be swapped