Name :- Pulkit Bansal
Section :- G
Branch :- C.S.E (core)
Roll No. :- 30
University Roll No. :- 2014789.

Tutorial Sheet-3

Ans ① to Ans ⑥

↓

already done in Assignment-01.

Ans ⑦ Program to find two index such is at
$A[i] + A[j] = k$

```
int main()
{
    int n, key;
    bool flag = false;
    {
        cin >> n;
    }
     cin >> key
    unmap < int, int > mp;
    for( int i = 0; i < n; i++)
    {
        int temp = key - v[i]
        if( mp.find(temp) == mp.end() )
        {
```

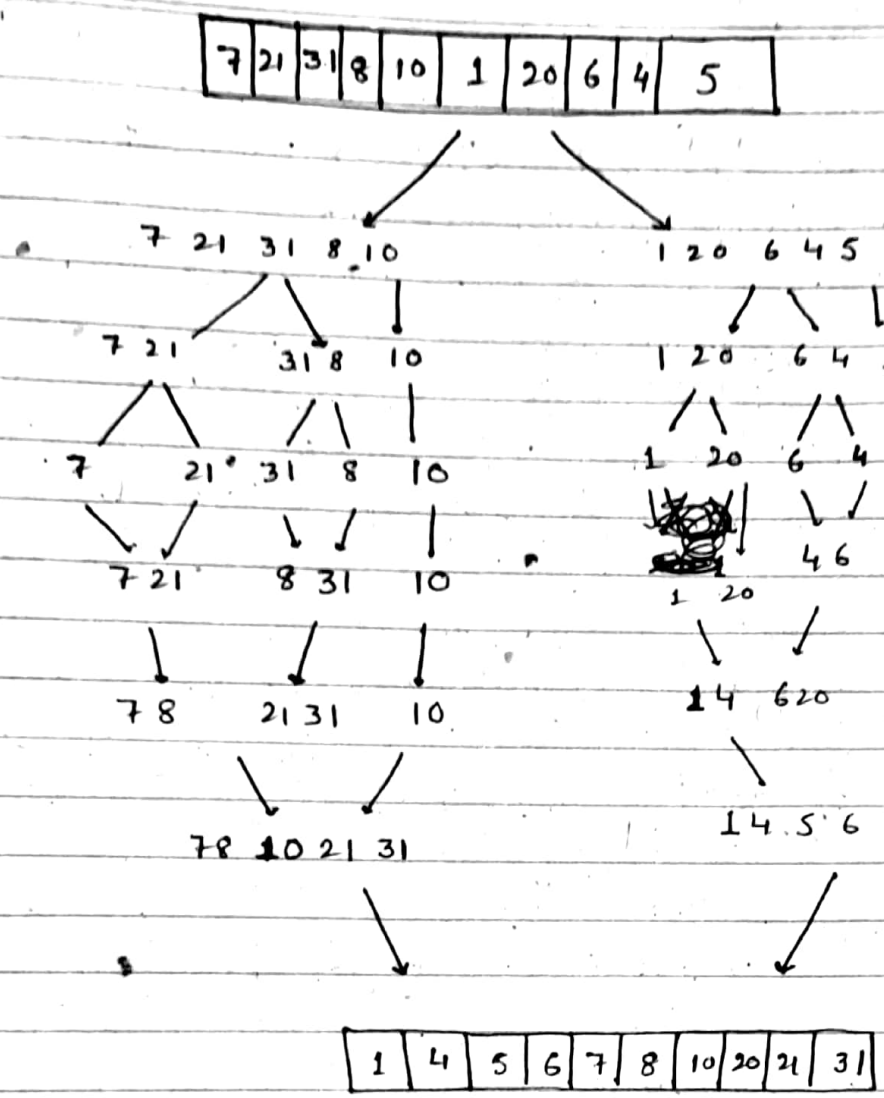
```
        mp[v[i]] = i;
    }
    else
    {
        cout << i << " " << mp[n];
        Flag = True;
        break;
    }
}
if (flag == false)
{
    cout << "No. sum. pair exist";
}
return 0;
}
```

Ans ⑧ Quick sort is the fastest general-purpose sort. In most practical situations, quick sort is the record of cutem.
→ If stabilities is important and space is available, merge sort might be best.

Ans ⑨ Inversion count for an array indicates how far (or close) the array is from being sorted. If the array is already sorted, then the inversion count is 0, but if the array is sorted in reverse order, the inversion count is the maximum.

Array arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5}.

for given array table. no of inversion = 31.



Ans (10) The Best case for quick sort will be when the middle element is picked as pivot
The worst case for quick sort is when array is sorted is either increasing or decreasing order.

Ans (11) Recurrence Relation

Best case	Worst Case
Quick sort : $T(n) = 2T(n/2) + n$	Quick sort : $T(n) = T(n-1) + n$
Merge sort : $T(n) = 2T(n/2) + n$	Merge sort : $T(n) = 2T(n/2) + n$

Similarities:-

① Both the method follow divide and conquer approach.
② Both have Best case Time complexity $O(n \log n)$.

Dibference:-

① Merge sort is a stable algorithm where quick sort sort is not stable sorting algorithm.
② Worst case T.C of quick sort is $O(n^2)$ where merge sort is $O(n \log n)$.

Any ②

```
void selection sort (int arr[], int n)
{
    for( int i=0 ; i<n-1 ; i++)
    {
        int min = i;
        for( int j = i+1; j<n; j++)
        {
            if ( arr.[min]. > arr [ j ])
            {
                min=j ;
            }
        }
    int key = arr [min];
    while (min>i)
    {
        arr [min]= arr [min-1]
        min --;
    }
    arr [i]= key
    }
}
```

Ans (13)

```
void bubble sort (int arr [], int n)
{
    int i, j;
    bool swapped;
    for(i=0; i<n-1; i++)
    {
        swapped = False;
        for( j=0; j<n-i-1; j++)
        {
            swap (arr[j], arr[j+1];
            swapped = True;
        }
    }
    if (swapped == False)
    {
        break;
    }
}
}
```

Ans (14) For this purpose we will use external sorting, technique. eg → Merge sort.

- In internal sorting all the data is stored in main memory all the time while sorting.
- In external sorting data is stored in the slower external memory (usually a hard disk). In the sorting phase of data small enough to bit in main memory are read, sorted and written out in a temporary file.