Name :- YASH CHANDRA

Section :- E

Branch :- CS

Roll No :- 62 / 2014954

Sheet :- Tut - 3

Ans-1 already done in Assignment -01

Ans-2 already done in Assignment - 01

Ans-3 already done in Assignment -02

Ans-4 already done in Assignment - 01

Ans-5 already done in Assignment · 01

Ans-6 already done in Assignment - 01

Ans-7 program to find two index such that

A [i] + A [j] = k

```
int main ()
{
  int n ,key;
  bool flag = False ;
 {
   cin >> n

 }
  cin >> key
 umap <int, int> mp;
 for (int i = 0; i<n ; i++)
   {
```

Yash

```
    int Temp = Key - v[i]
    if (mp.find(Temp) == mp.end())
     {
        mp [v[i]] = i;
      }
    else.
     {
      cout .<< i << " "<< mp [x];
         Flag = True.;
          break;
      }
  }
   if .(flag == false)
   {
      cout .<< "No such pair exist";
   }
  return 0;
}
```

Ans - 8

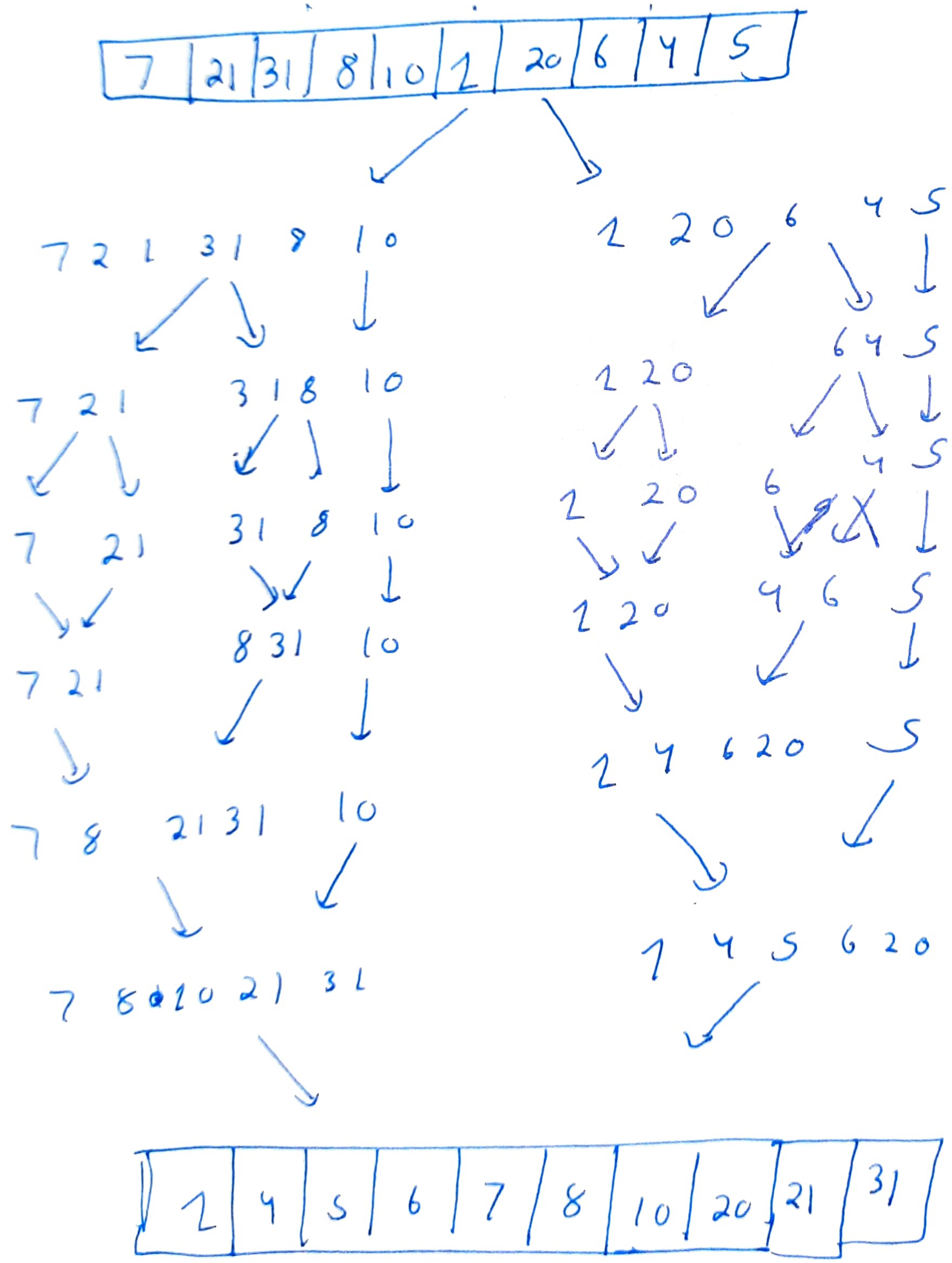
Quick sort is a. the fastest General -purpose sort. In most practical situations, quick sort is the method of choice

→ If stability 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 case no of inversions = 31



Ans-10 The Best case for Quick sort will be when the middle element is picked as a pivot

The worst case for Quick sort is when array is sorted in either increasing or decreasing order.

Ans-11 Recurrence Relation

Best case

Quick sort = $T(n) = 2T(n/2) + n$

merge sort = $T(n) = 2T(n/2) + n$

Worst case

Quick sort

$T(n) = T(n-1) + n$

merge sort

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

Similarities :-

① Both the methods follow divide and conquer Approach.

② Both have Best case Time complexity $O(n \log n)$

Difference

① merge sort is a stable algorithm where quick sort is not stable sorting algorithm.

② Worst case T.C of quick sort is $O(n^2)$ where merge sort is $O(n \log n)$

Ans-12

```
void selection sort (int arr[], int n)
{
```

Yash

```
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-17 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 drives external memory (usually a hard disk). In the sorting phase, chunks of data small enough to fit in the main memory are read, sorted and written out to a temporary file.

4 abs