Name - Shantanu Deshpande
Class - SYCSE
Roll no. - 7
Batch - S1
Sub. - D.S

Practical No. 9

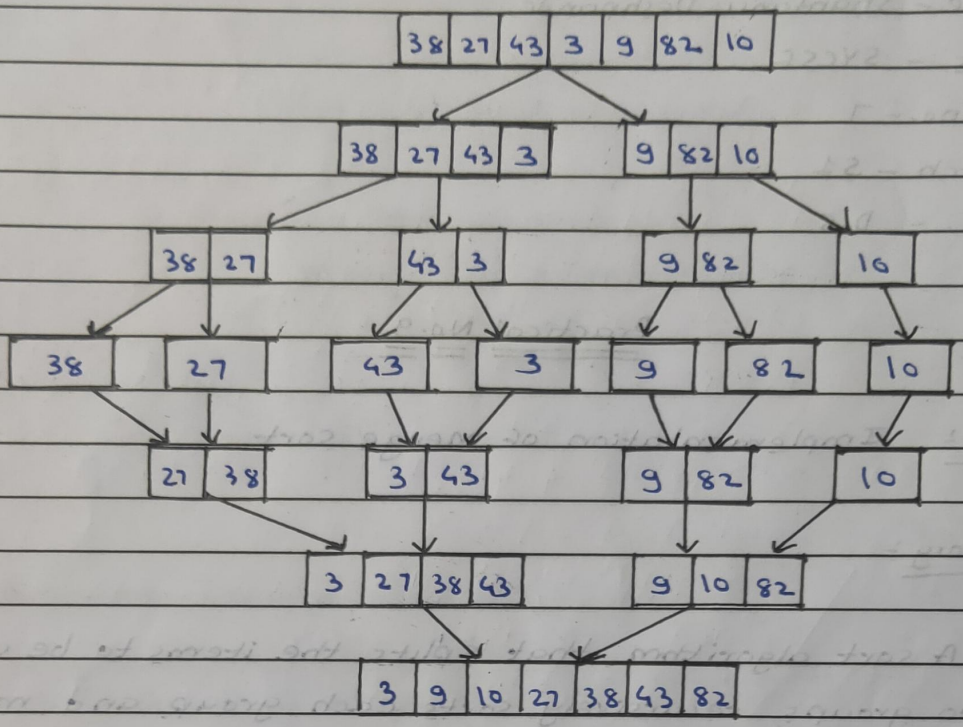
Aim: Implementation of Merge sort

Theory:

A sort algorithm that splits the items to be sorted into two groups, recursively sorts each group, and merges them into a final, sorted sequence.

To sort an array of n elements, we perform the following three steps in sequence:

- If n < 2, then the array is already sorted. Stop now.
- Otherwise, n > 1 and we perform the following three steps in sequence:
  1. Sort the left half of the array.
  2. Sort the right half of the array.
  3. Merge the now-sorted left and right halves.



Algorithm :-

```
Algorithm Mergesort (low, high)
  // a[low : high] is a global array to be sorted.
  // Small(P) is true if there is only one element
  // to sort. In this case the list is already sorted
  {
    if (low < high) then // If there are more than one element
    {
      // Divide P into subproblems.
          // Find where to split the set.
          mid := [(low + high)/2];
```

```
    // Solve the subproblems.
        Merge sort (low, mid);
        Merge sort (mid+1, high);

    // Combine the solutions.
        Merge (low, mid, high);

  }
}
```

Algorithm Merge (low, mid, high);

```
// a [low:high] is a global array containing two sorted
// subsets in a[low:mid] and in a [mid+1:high]. The goal
// is to merge these two sets into a single set residing
// in a [low:high]. b[] is an auxillary global array.

{
  h := low; i := low; j := mid+1;
  while ((h ≤ mid) and (j ≤ high)) do
  {
    if (a[h] ≤ a[j]) then
    {
      b[i] := a[h]; h := h+1;
    }
    else
```

```
    {
      b[i] := a[j] ; j := j+1;
    }
    i := i+1;
}
if (h>mid) then
    for k := j to high do
    {
      b[i] := a[k]; i := i+1;
    }
else
    for k := h to mid do
    {
      b[i] := a[k]; i = i+1;
for k := low to high do a[k] := b[k];
}
```

Conclusion :- Thus we have implemented Merge sort.

Name- Shantanu Deshpande
Class- SYCSE
Roll no.-7
Batch-S1

```
#include<stdio.h>
int a[20] ,b[20];
void merge(int low,int mid,int high)
{
int h=low,j=mid+1, i=low,k;
while( h<=mid &&j<=high)
{

if(a[h] < a[j])
{ b[i]= a[h]; h++;
}
else
{ b[i]= a[j]; j++;
}
i++;

}
if (h >mid)
{
for(k=j; k<=high;k++)

{ b[i] =a[k]; i++;}

}
else
{
for(k=h; k<=mid;k++)

{ b[i] =a[k]; i++;}

}
for(k=low; k<=high;k++)
a[k] =b[k];

}

void mergesort(int low, int high)
{
int mid;
if(low < high)
{

mid =(low+high)/2;
mergesort(low,mid);
mergesort(mid+1, high);
merge(low,mid,high);

}
}
```

```
void main()
{
int n,i;
printf("Enter The total no of elements");
scanf("%d", &n);
printf("enter Elements:");
for(i=1;i<=n ;i++)
scanf( "%d",&a[i]);
mergesort(1,n);
printf("Sorted elements are:\n");
for(i=1;i<=n ;i++)
printf( "%d ",a[i]);
}
```

GDB online Debugger | Compiler

onlinegdb.com

Run | Debug | Stop | Share | Save | { } Beautify | Language C

main.c

input

```
Enter The total no of elements7
enter Elements:56
75
25
67
89
48
57
Sorted elements are:
25 48 56 57 67 75 89

...Program finished with exit code 0
Press ENTER to exit console.
```