PRACTICAL-04

//WAP to implement merge sort:

ALGORITHM—

1. MERGE_SORT(arr, beg, end)
2. **if** beg < end
3. set mid = (beg + end)/2
4. MERGE_SORT(arr, beg, mid)
5. MERGE_SORT(arr, mid + 1, end)
6. MERGE (arr, beg, mid, end)
7. end of **if**
8. END MERGE_SORT

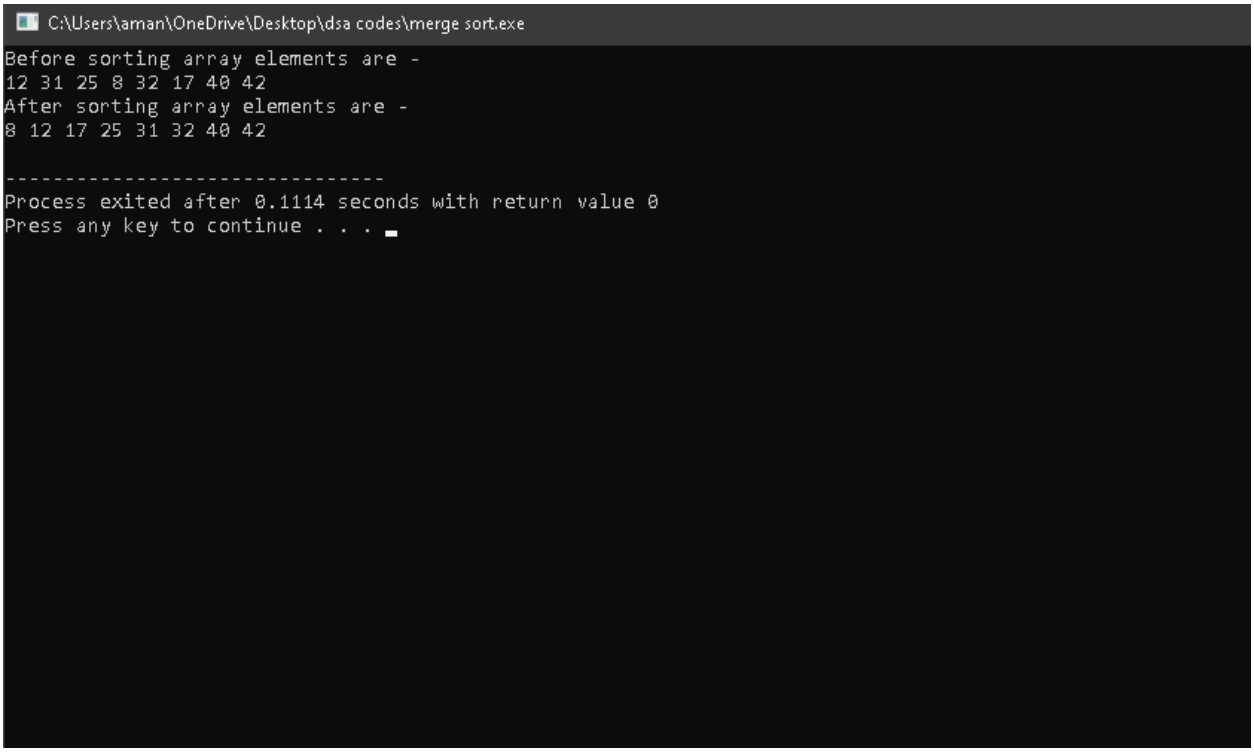
//Code

```
#include <stdio.h>
void merge(int a[], int beg, int mid, int end)
{
int i, j, k;
int n1 = mid - beg + 1;
int n2 = end - mid;

int LeftArray[n1], RightArray[n2];
for (int i = 0; i < n1; i++)
LeftArray[i] = a[beg + i];
for (int j = 0; j < n2; j++)
RightArray[j] = a[mid + 1 + j];
```

```
i = 0;
j = 0;
k = beg;

while (i < n1 && j < n2)
{
if(LeftArray[i] <= RightArray[j])
{
a[k] = LeftArray[i];
i++;
}
else
{
a[k] = RightArray[j];
j++;
}
k++;
}
while (i<n1)
{
a[k] = LeftArray[i];
i++;
k++;
}

while (j<n2)
{
a[k] = RightArray[j];
j++;
k++;
}
}
```

```
void mergeSort(int a[], int beg, int end)
{
if (beg < end)
{
int mid = (beg + end) / 2;
mergeSort(a, beg, mid);
mergeSort(a, mid + 1, end);
merge(a, beg, mid, end);
}
}
void printArray(int a[], int n)
{
int i;
for (i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
}

int main()
{
int a[] = { 12, 31, 25, 8, 32, 17, 40, 42 };
int n = sizeof(a) / sizeof(a[0]);
printf("Before sorting array elements are - \n");
printArray(a, n);
mergeSort(a, 0, n - 1);
printf("After sorting array elements are - \n");
printArray(a, n);
return 0;
}
```

output


C:\Users\aman\OneDrive\Desktop\dsa codes\merge sort.exe
Before sorting array elements are -
12 31 25 8 32 17 40 42
After sorting array elements are -
8 12 17 25 31 32 40 42

Process exited after 0.1114 seconds with return value 0
Press any key to continue . . .