WEEK-4 MODULE

NAME:MD.Amrath **ID**:B181094 **CLASS**:AB2-305

1.BUBBLE SORT

PSEUDOCODE:

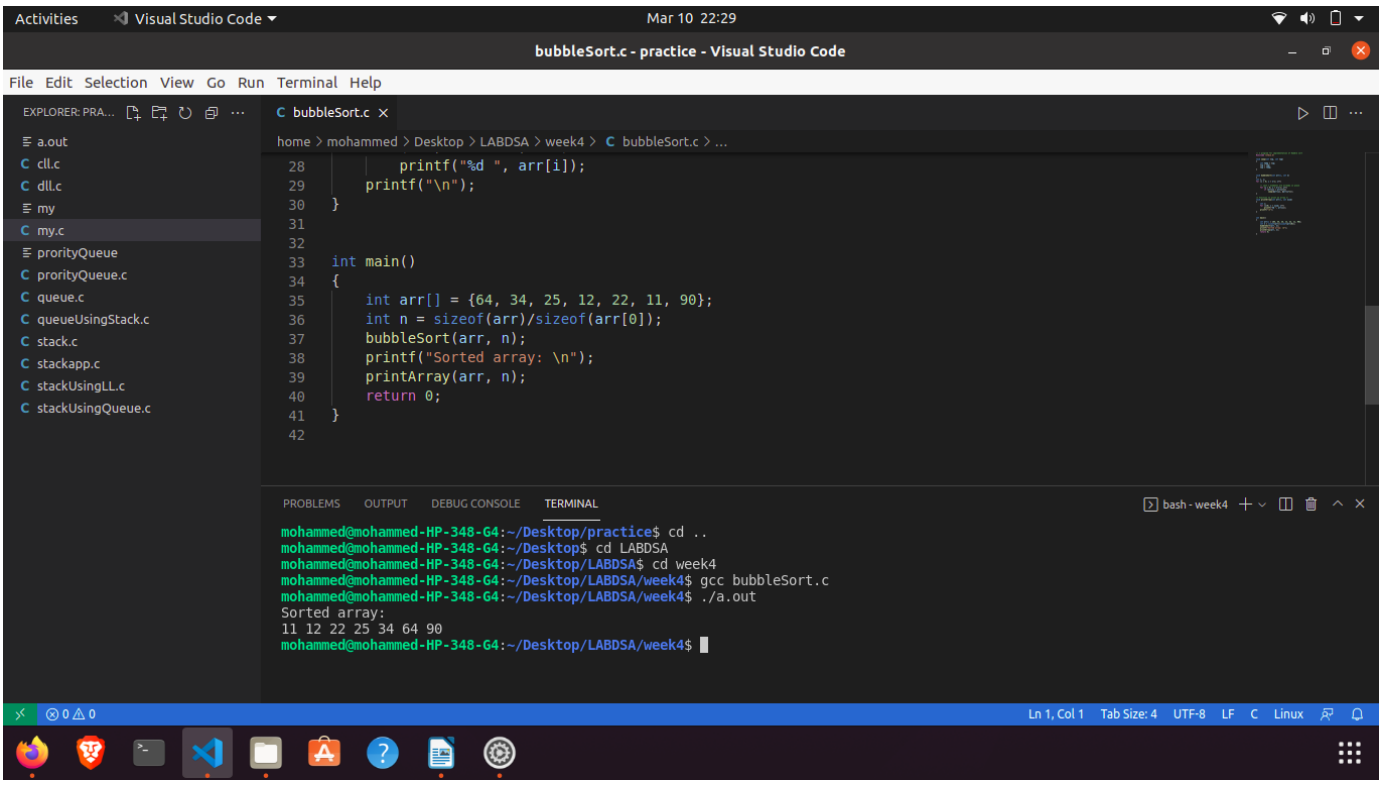
```
procedure bubbleSort( list : array of items )

   loop = list.count;

   for i = 0 to loop-1 do:
      swapped = false

      for j = 0 to loop-1 do:

         /* compare the adjacent elements */
         if list[j] > list[j+1] then
            /* swap them */
            swap( list[j], list[j+1] )
            swapped = true
         end if

      end for

      /*if no number was swapped that means
      array is sorted now, break the loop.*/

      if(not swapped) then
         break
      end if

   end for

end procedure return list
```

CODE:

```
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
        int temp = *xp;
        *xp = *yp;
        *yp = temp;
}


void bubbleSort(int arr[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)
```

```
        // Last i elements are already in place
        for (j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                        swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
        int i;
        for (i=0; i < size; i++)
                printf("%d ", arr[i]);
        printf("\n");
}


int main()
{
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        int n = sizeof(arr)/sizeof(arr[0]);
        bubbleSort(arr, n);
        printf("Sorted array: \n");
        printArray(arr, n);
        return 0;
}
```

OUTPUT:



2.NISERTION SORT

PSEUDOCODE:

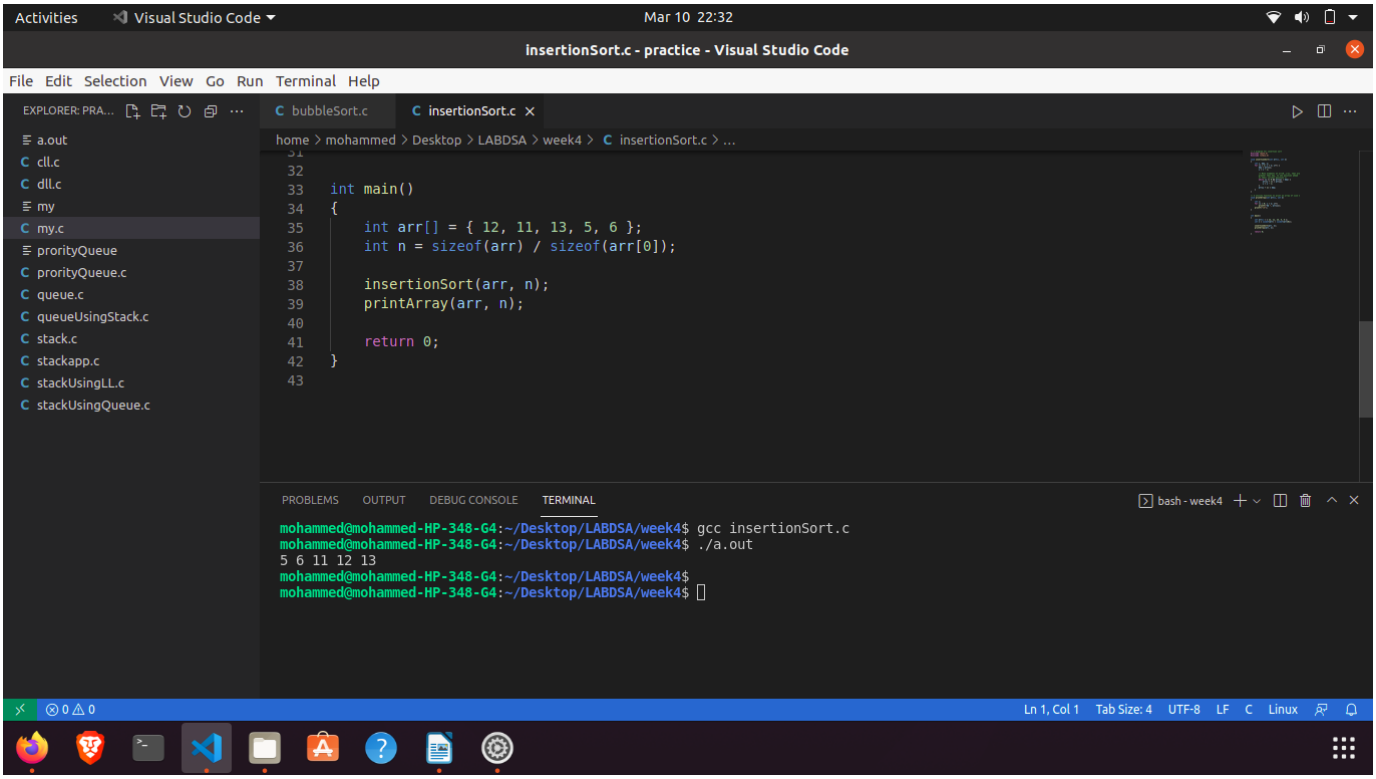
To sort an array of size n in ascending order:
1: Iterate from arr[1] to arr[n] over the array.
2: Compare the current element (key) to its predecessor.
3: If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

CODE:

```
// C program for insertion sort
#include <math.h>
#include <stdio.h>

void insertionSort(int arr[], int n)
{
	int i, key, j;
	for (i = 1; i < n; i++) {
		key = arr[i];
		j = i - 1;

		/* Move elements of arr[0..i-1], that are
		greater than key, to one position ahead
		of their current position */
		while (j >= 0 && arr[j] > key) {
			arr[j + 1] = arr[j];
			j = j - 1;
		}
		arr[j + 1] = key;
	}
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
	int i;
	for (i = 0; i < n; i++)
		printf("%d ", arr[i]);
	printf("\n");
}


int main()
{
	int arr[] = { 12, 11, 13, 5, 6 };
	int n = sizeof(arr) / sizeof(arr[0]);

	insertionSort(arr, n);
	printArray(arr, n);
```

```
    return 0;
}
```

OUTPUT:



3.SELECTION SORT:

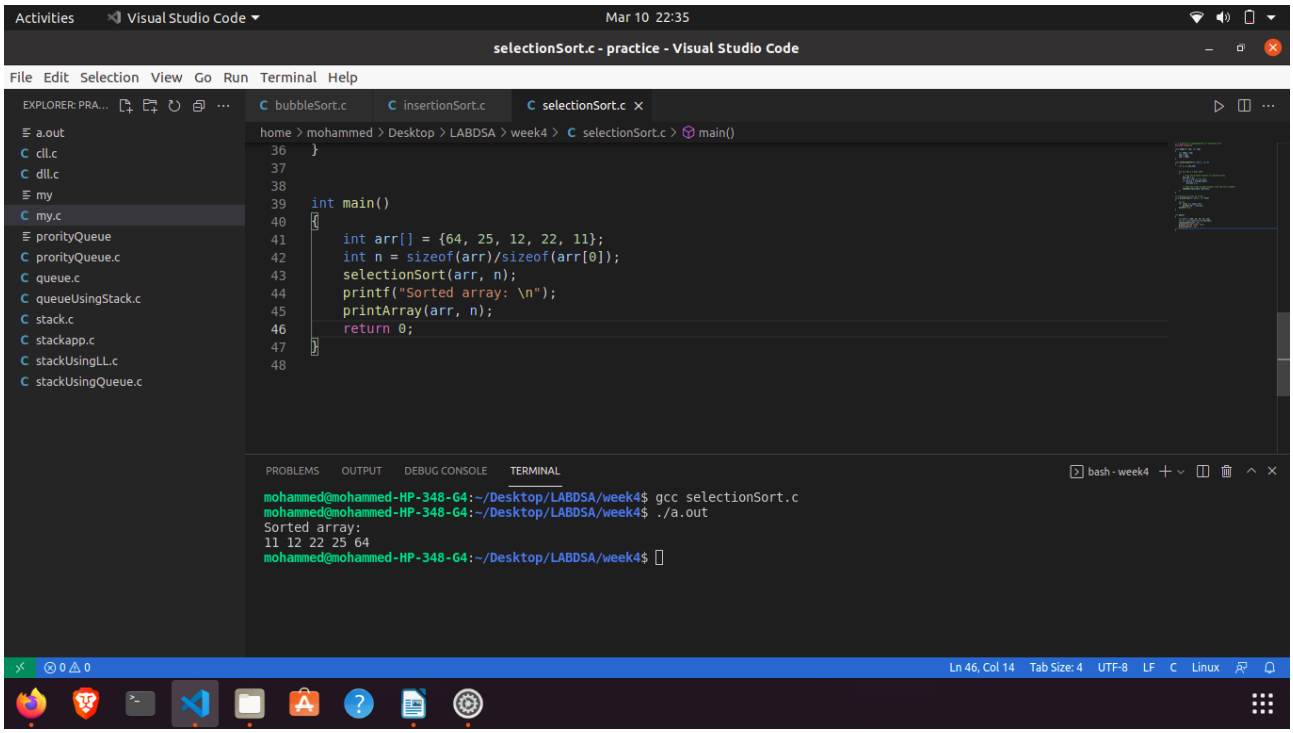
PSEUDOCODE:
The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.
1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.
In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

CODE:

```
// C program for implementation of selection sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
```

```
void selectionSort(int arr[], int n)
{
	int i, j, min_idx;

	/
	for (i = 0; i < n-1; i++)
	{
		// Find the minimum element in unsorted array
		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;

		// Swap the found minimum element with the first element
		swap(&arr[min_idx], &arr[i]);
	}
}

/* Function to print an array */
void printArray(int arr[], int size)
{
	int i;
	for (i=0; i < size; i++)
		printf("%d ", arr[i]);
	printf("\n");
}


int main()
{
	int arr[] = {64, 25, 12, 22, 11};
	int n = sizeof(arr)/sizeof(arr[0]);
	selectionSort(arr, n);
	printf("Sorted array: \n");
	printArray(arr, n);
	return 0;
}
```

OUTPUT:



4.QUICKSORT:

PSEUDOCODE:

```
function partitionFunc(left, right, pivot)
   leftPointer = left
   rightPointer = right - 1

   while True do
      while A[++leftPointer] < pivot do
         //do-nothing
      end while

      while rightPointer > 0 && A[--rightPointer] > pivot do
         //do-nothing
      end while

      if leftPointer >= rightPointer
         break
      else
         swap leftPointer,rightPointer
      end if

   end while

   swap leftPointer,right
   return leftPointer

end function

procedure quickSort(left, right)

   if right-left <= 0
      return
```

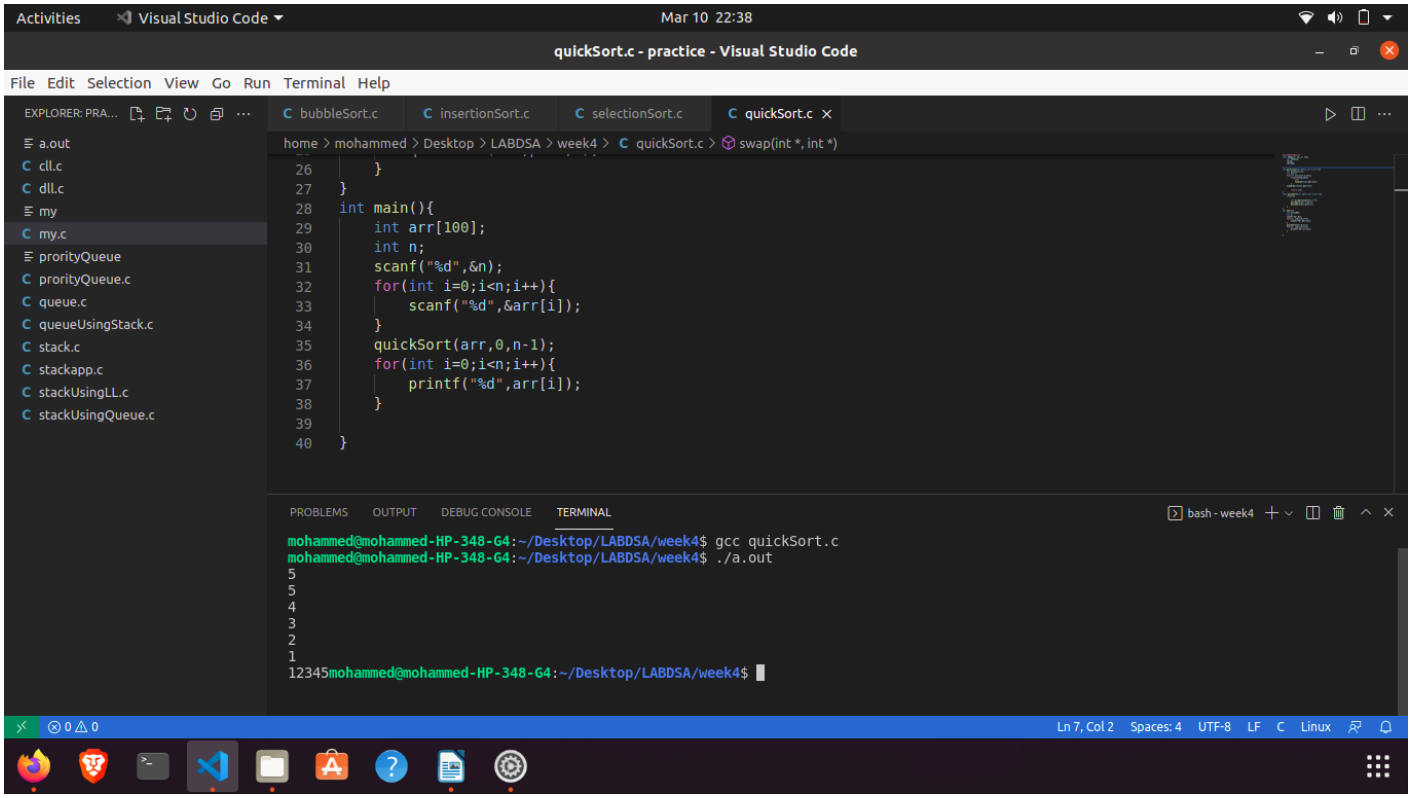
```
   else
      pivot = A[right]
      partition = partitionFunc(left, right, pivot)
      quickSort(left,partition-1)
      quickSort(partition+1,right)
   end if

end procedure
```

CODE:

```
#include<stdio.h>
void swap(int *a,int *b){
   int temp=*a;
   *a=*b;
   *b=temp;

}
int partition(int arr[],int l,int r){
   int pivot=arr[r];
   int i=l-1;
   for(int j=l;j<=r-1;j++){
      if(arr[j]<pivot){
         i++;
         swap(&arr[i],&arr[j]);
      }
   }swap(&arr[i+1],&arr[r]);

      return i+1;
}
void quickSort(int arr[],int l,int r){
   if(l<r){

      int pi=partition(arr,l,r);
      quickSort(arr,l,pi-1);
      quickSort(arr,pi+1,r);
   }
}
int main(){
   int arr[100];
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   quickSort(arr,0,n-1);
   for(int i=0;i<n;i++){
      printf("%d",arr[i]);
   }
```

}

OUTPUT:



4.MERGESORT:
PSEUDOCODE:

We shall now see the pseudocodes for merge sort functions. As our algorithms point out two main functions − divide & merge.

Merge sort works with recursion and we shall see our implementation in the same way.

```
procedure mergesort( var a as array )
   if ( n == 1 ) return a

   var l1 as array = a[0] ... a[n/2]
   var l2 as array = a[n/2+1] ... a[n]

   l1 = mergesort( l1 )
   l2 = mergesort( l2 )

   return merge( l1, l2 )
end procedure

procedure merge( var a as array, var b as array )

   var c as array
   while ( a and b have elements )
```

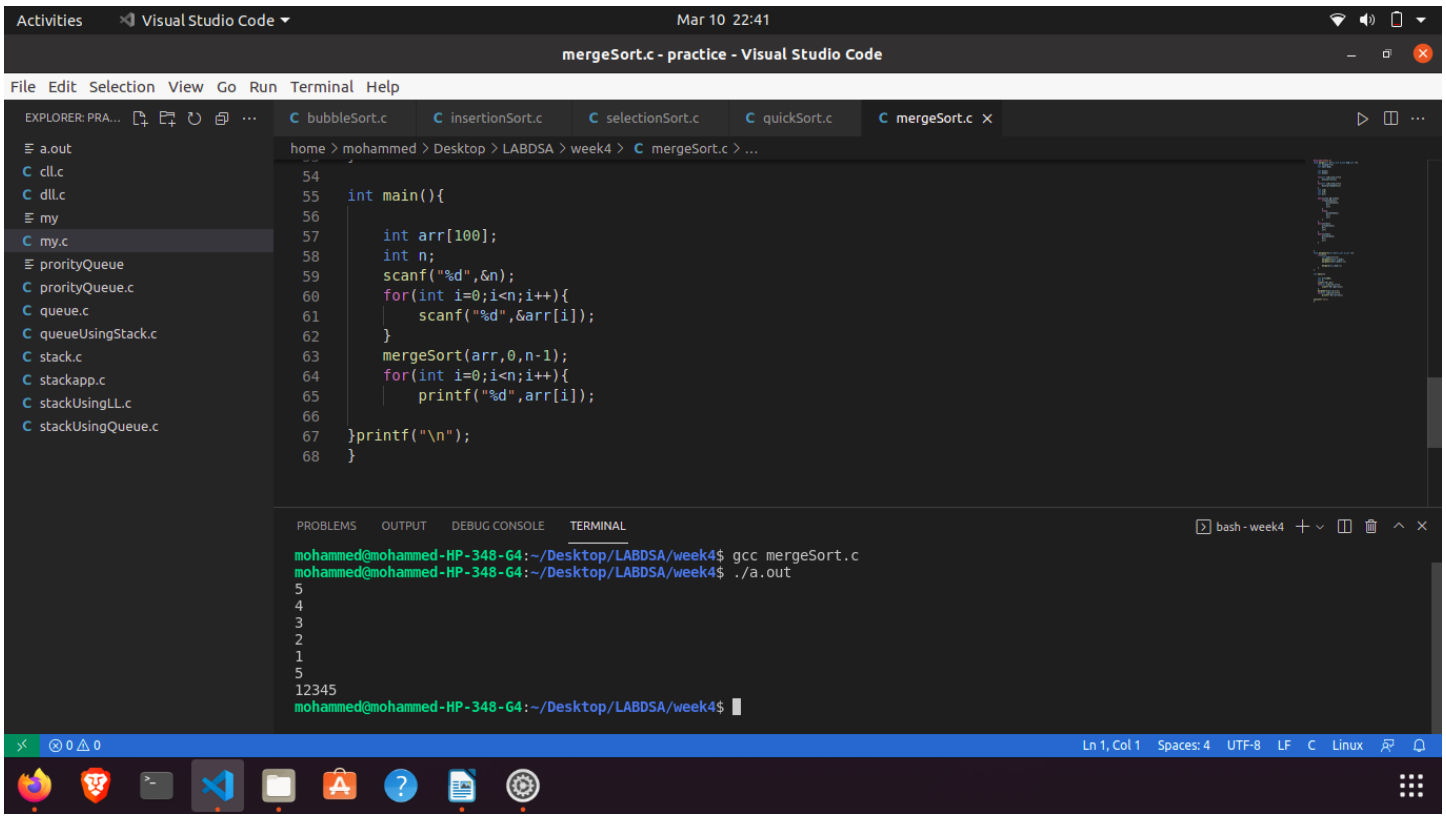
```
      if ( a[0] > b[0] )
         add b[0] to the end of c
         remove b[0] from b
      else
         add a[0] to the end of c
         remove a[0] from a
      end if
   end while

   while ( a has elements )
      add a[0] to the end of c
      remove a[0] from a
   end while

   while ( b has elements )
      add b[0] to the end of c
      remove b[0] from b
   end while

   return c

end procedure
```

CODE:

```
#include<stdio.h>
void merge(int arr[],int l,int mid,int r){
   int n1=mid-l+1;
   int n2=r-mid;

   int a[n1];
   int b[n2];

   for(int i=0;i<n1;i++){
      a[i]=arr[l+i];
   }
   for(int i=0;i<n2;i++){
      b[i]=arr[mid+1+i];
   }
   int j=0;
   int i=0;
   int k=l;

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

```
    while(i<n1){
        arr[k]=a[i];
        k++;
        i++;
    }
    while(j<n2){
        arr[k]=b[j];
        k++;
        j++;
    }



}
void mergeSort(int arr[],int l,int r){
    if(l<r){
        int mid=(l+r)/2;
        mergeSort(arr,l,mid);
        mergeSort(arr,mid+1,r);

        merge(arr,l,mid,r);
    }
}

int main(){

    int arr[100];
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    mergeSort(arr,0,n-1);
    for(int i=0;i<n;i++){
        printf("%d",arr[i]);

}printf("\n");
}
```

OUTPUT:


Activities
Visual Studio Code
Mar 10 22:41
mergeSort.c - practice - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER: PRA...
a.out
cll.c
dll.c
my
my.c
prorityQueue
prorityQueue.c
queue.c
queueUsingStack.c
stack.c
stackapp.c
stackUsingLL.c
stackUsingQueue.c
bubbleSort.c
insertionSort.c
selectionSort.c
quickSort.c
mergeSort.c
home > mohammed > Desktop > LABDSA > week4 > mergeSort.c > ...
54
55 int main(){
56
57 int arr[100];
58 int n;
59 scanf("%d",&n);
60 for(int i=0;i<n;i++){
61 scanf("%d",&arr[i]);
62 }
63 mergeSort(arr,0,n-1);
64 for(int i=0;i<n;i++){
65 printf("%d",arr[i]);
66
67 }printf("\n");
68 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - week4
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week4$ gcc mergeSort.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week4$ ./a.out
5
4
3
2
1
5
12345
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week4$
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
C
Linux