~~Worksheet 1.3~~

Student Name: Garv Khurana **UID: 21BCS6615**
Branch: CSE - AIML **Section/Group-: 21AML - 9 - "A"**
Semester: 3rd
Subject Name: Data Structures **Subject Code:** 21CSH - 241

1. Aim:

A) **Program to sort a given array using a quick sort**
B) **Program to sort a given array using a merge sort**
C) **Program to sort a given array using a selection sort**
D) **Program to sort a given array using a bubble sort**

2. Algoritm:

a) **Step1:** Start
Step2: Creat a function for swaping a number.
Step3: Creat a function for choosing a pivote element.
Step4: Creat a function for using quick sort.
Step5: Input the size of the array from the user.
Step6: Input the array from the user.
Step7: Call a sort function.
Step8: Print a sorted array.
Step9: Stop

b) **Step1:** Start
Step2: Creat a function to use merge sort
Step3: Creat a function for merging the two sorted array in sorted order
Step4: Creat a function for print a sorted array
Step5: Input the size of the array from the user.
Step6: Input the array from the user.
Step7: Call a sort function.
Step8: Call a merge function
Step9: Call a print function to print a sorted array
Step10: Stop

c) **Step 1:** Start
Step 2: Input the size of the array in n
Step 3: Intialize the array of size n
Step 4: Input the element of the array
Step 5: Apply the procedure for selection sort
Step 6: Print the sorted array
Step 7: Stop

d) **Step 1:** Start
Step 2: Input the size of the array in n
Step 3: Intialize the array of size n
Step 4: Input the element of the array

Step 5: Apply the procedure for bubble sort
Step 6: Print the sorted array
Step 7: Stop

3. Code:

a)

```
#include<stdio.h>
void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}
int partition(int *arr, int p, int r)
{
   int x = arr[r];
   int i = p - 1;
   for (int j = p; j < r; j++)
   {
      if (arr[j] <= x)
      {
         i++;
         swap(&arr[i],&arr[j]);
      }
   }
   swap(&arr[i + 1],&arr[r]);
   return i + 1;
}

void Sort(int *arr, int p, int r)
{
   if (p < r)
   {
      int q = partition(arr, p, r);
      Sort(arr, p, q - 1);
      Sort(arr, q + 1, r);
   }
}
int main()
{
   int n;
   printf("size of the arrray-: ");
   scanf("%d",&n);
   int arr[n];
   for (int i = 0; i < n; i++)
   {
      scanf("%d",&arr[i]);
   }
   Sort(arr, 0, n);
   printf("Sorted Array: \n");
```

```
            for (int i = 0; i < n; i++)
            {
               printf("%d  ",arr[i]);
            }
         }
```

b)

```
        #include<stdio.h>
        void merge(int *arr,int left,int mid,int right){
        int *ptr=(int*) malloc((right+1)*sizeof(int));
        int i=left;
        int j=mid+1;
        int k=left;
        while(i<=mid&&j<=right){
         if(arr[i]>arr[j]){
         ptr[k]=arr[j];
         ++j;
        ++k;
    }
    else if(arr[i]<=arr[j]){
        ptr[k]=arr[i];
        ++i;
        ++k;
    }
  }
  while(i<=mid||j<=right){
    if(i<=mid){
        ptr[k]=arr[i];
        ++i;
        ++k;
    }
    else if(j<=right){
        ptr[k]=arr[j];
        ++j;
        ++k;
    }
  }
  for(i=left;i<=right;++i){
    arr[i]=ptr[i];
  }
  free(ptr);
  ptr=NULL;
}
void mergesort(int *arr,int left,int right){
  if(left<right){
    int mid=(left+right)/2;
    mergesort(arr,left,mid);
    mergesort(arr,mid+1,right);
    merge(arr,left,mid,right);
  }
}
void printing(int *arr,int right){
  for(int i=0;i<=right;++i){
    printf("\t%d",arr[i]);
  }
  printf("\n");
}
int main(){
  int n;
  printf("enter the size of the array -: ");
  scanf("%d",&n);
  int arr[n];
  for(int i=0;i<n;i++)
  {
    scanf("%d",&arr[i]);
  }
  int left=0;
  int right=n-1;
   printing(arr,right);
  mergesort(arr,left,right);
  printing(arr,right);
  return 0;
}
```

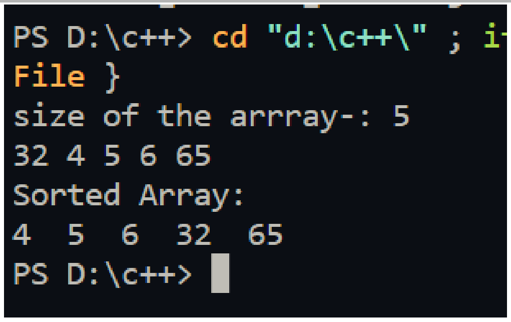
C)

```
#include<stdio.h>
int main()
{
    int n;
    int min=0;
    printf("enter the size of the element: ");
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(arr[i]>arr[j])
            {
                int temp = arr[j];
                arr[j] = arr[i];
                arr[i] = temp;
            }
        }
    }
    printf("sorted arry is: ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",arr[i]);
    }
}
```

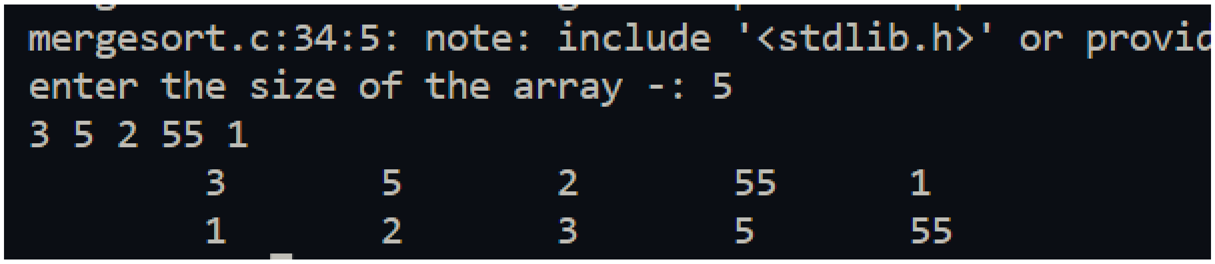
D)

```
#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++)
        for (j = 0; j < n - i - 1; j++)
        if (arr[j] > arr[j + 1])
        swap(&arr[j], &arr[j + 1]);
}
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;
}
```

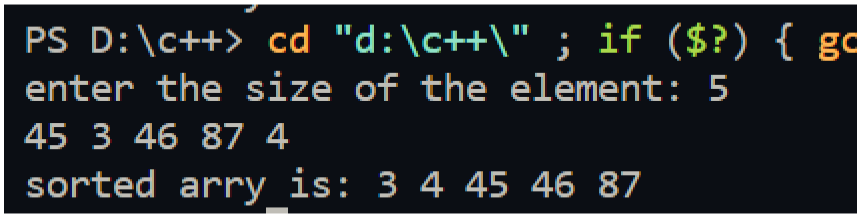
Output-:

a) 

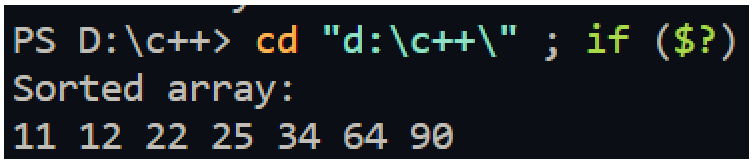
```
PS D:\c++> cd "d:\c++\" ; i
File }
size of the arrray-: 5
32 4 5 6 65
Sorted Array:
4  5  6  32  65
PS D:\c++> 
```

b) 

```
mergesort.c:34:5: note: include '<stdlib.h>' or provic
enter the size of the array -: 5
3 5 2 55 1
        3       5       2       55      1
        1       2       3       5       55
```

c) 

```
PS D:\c++> cd "d:\c++\" ; if ($?) { gc
enter the size of the element: 5
45 3 46 87 4
sorted arry is: 3 4 45 46 87
```

d) 

```
PS D:\c++> cd "d:\c++\" ; if ($?)
Sorted array:
11 12 22 25 34 64 90
```

Learning outcomes(What I have learnt):

1. **Learn the procedure of merge sort**
2. **Learn how to merge two sorted array.**
3. **Learn how to input an array.**
4. **Learn how to print a sorted array.**
5. **Learn the procedure of quick sort**

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			