Dr. D.Y.Patil Arts, Commerce and Science College,

Pimpri, Pune-18

S.Y.B.Sc(Comp. Sci) 2024-25

Data Structures and Algorithms – I

Practical Assignment 3: Sorting Algorithms – Counting Sort, Merge Sort, Quick Sort

Set A

a) Sort a random array of n integers (accept the value of n from user) in ascending order by using recursive Counting sort algorithm.

```
//Counting Sort
#include<stdio.h>
void display(int a[], int n);
int getMax(int a[],int n)
 {
      int max = a[0],i;
      for(i=1;i<n;i++) {
            if(a[i]>max)
                  max=a[i];
      }
      return max;
}

void countSort(int a[],int n)
{
      int output[n+1],i;
      int max=getMax(a,n);
      int count[max+1];
      for(i=0;i<=max;++i)
      {
            count[i]=0;
      }

      for (i=0;i<n;i++)
      {
            count[a[i]]++;
      }
         printf("\n count of each element:  ");
             display(count,max+1);
      for(i=1;i<=max;i++)
            count[i] =count[i]+count[i-1];

     printf("\nCumulative array:  ");
       display(count,max+1);

      for (i = n - 1; i >= 0; i--)
       {
            output[count[a[i]] - 1] = a[i];
            count[a[i]]--;
      }

      for(i = 0; i<n; i++)
            a[i] = output[i];
```

```
}
void display(int a[],int n)
{
      int i;
      for (i=0;i<n;i++)
            printf("%d  ",a[i]);
}
main()
{
      int a[10],i,n;
      printf("\nEnter number of elements: ");
      scanf("%d",&n);
      printf("\nEnter the unsorted numbers: ");
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);
      printf("Before sorting array elements are - ");
      display(a, n);
      countSort(a, n);
      printf("\nAfter sorting array elements are - ");
      display(a, n);
}
```

b)Sort a random array of n integers (accept the value of n from user) in ascending order by using a recursive Merge sort algorithm.

```
// Merge sort

#include<stdio.h>

void display(int a[],int n);
void merge(int a[],int low,int mid,int high);
void mergesort(int a[],int low,int high);
main()
{
      int a[20],i,n;
      printf("\nHow many numbers: ");
      scanf("%d",&n);
      printf("\nEnetr the unsorted numbers: ");
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);
      mergesort(a,0,n-1);
      printf("\n\nThe Sorted list is: ");
      display(a,n);
}
void display(int a[],int n)
{
      int i;
      printf("\n");
      for(i=0;i<n;i++)
            printf("\t%d",a[i]);
}
void merge(int a[],int low,int mid,int high)
{
      int i,j,k,b[20];
```

```
      i=low;
      j=mid+1;
      k=0;
      while ((i<=mid) && (j<=high))
      {
            if (a[i]<=a[j])
                  b[k++]=a[i++];
            else

                  b[k++]=a[j++];

      }
      while (i<=mid)
      {

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

      }
      while (j<=high)
      {
            b[k++]=a[j++];

      }
      //copy merged element from b to a
      for(j=low,k=0;j<=high;j++,k++)
      {

            a[j]=b[k];
      }
}
void mergesort(int a[],int low,int high)
{
      int mid;
      if(low<high)
      {
            mid=(low+high)/2;
            mergesort(a,low,mid);
            mergesort(a,mid+1,high);
            merge(a,low,mid,high);
      }
}
/*
 [root@localhost setA]# cc merge.c
[root@localhost setA]# ./a.out

How many numbers: 5

Enetr the unsorted numbers: 3 6 1 2 4


The Sorted list is:
      1     2     3     4     6
*/
```

c) Sort a random array of n integers (create a random array of n integers) in ascending order by using recursive Quick sort algorithm.

```
//  Quick Sort
#include<stdio.h>
void display(int a[],int n);
int partition(int a[],int lb,int ub);
void quicksort(int a[],int lb,int ub);
main()
{
      int a[20],i,n;
      printf("\nEnter number of elements: ");
      scanf("%d",&n);
      printf("\nEnter the unsorted numbers: ");
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);
      quicksort(a,0,n-1);
      printf("\nThe sorted list: ");
      display(a,n);
}
void display(int a[],int n)
{
      int i;
      printf("\n");
      for(i=0;i<n;i++)
            printf("\t%d",a[i]);
}
int partition(int a[],int lb,int ub)
{
      int up,down;
      int temp,pivot;
      up=ub;
      down=lb+1;
      pivot=a[lb];
      do
      {
            while((a[down]<pivot) && (down<=ub))
                  down++;
            while((a[up]>pivot) && (up>lb))
                  up--;

            if(down<up)
            {
                  temp=a[down];
                  a[down]=a[up];
                  a[up]=temp;
            }
      }while(down<up);
      //interchange pivot and a[up]
      a[lb]=a[up];
      a[up]=pivot;
      return up;
}
void quicksort(int a[],int lb,int ub)
{
```

```
      int j;
      if(lb<ub)
      {
            j=partition(a,lb,ub);
            quicksort(a,lb,j-1);
            quicksort(a,j+1,ub);
      }
}
/*
 [root@localhost ass3]# cc quick.c
[root@localhost ass3]# ./a.out

Enter number of elements: 5

Enter the unsorted numbers: 3 2 4 5 1

The sorted list:
      1     2     3     4     5
*/
```

Set B

a)Read the data from the ‘employee.txt’ file and sort on age using Counting sort, Merge sort, Quick sort and write the sorted data to another file 'sortedemponage.txt'.

b)Read the data from the file and sort on names in alphabetical order (use strcmp) using Counting sort, Merge sort, Quick sort and write the sorted data to another file 'sortedemponname.txt'.