**WEEK III**

1. **Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts ( shifts - total number of times the array elements are shifted from their place) required for sorting the array.**

**ANSWERS I. PROGRAM**

#include<stdio.h>

#define MAX 100

void insertion\_sort(int A[] ,int n)

{

int comparisons=0 ,shifts=0;

int i ,j ,temp;

for(i=1 ;i<n ;i++)

{

temp=A[i];

j=i-1;

shifts++;

while(j>=0 && temp<=A[j])

{

comparisons++;

A[j+1]=A[j];

j--;

shifts++;

}

A[j+1]=temp;

}

printf("Sorted Array :");

for(int i=0 ;i<n ;i++)

{

printf("%d\t",A[i]);

}

printf("\nTotal shifts %d and comparisons are %d",shifts ,comparisons);

}

int main()

{

int t ,n ,A[MAX];

printf("Enter the test cases :");

scanf("%d",&t);

while(t--)

{

printf("\nEnter the size of the array :");

scanf("%d",&n);

printf("Enter the elements in the array :");

for(int i=0 ;i<n ;i++)

{

scanf("%d",&A[i]);

}

insertion\_sort(A,n);

}

return 0;

}

**OUTPUT**

Enter the test cases :1

Enter the size of the array :8

Enter the elements in the array :12 0 98 34 78 23 32 76

Sorted Array :0 12 23 32 34 76 78 98

Total shifts 18 and comparisons are 11

1. **Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.**

**ANSWER III. PROGRAM**

#include<stdio.h>

#define MAX 100

void selection\_sort(int A[] ,int n)

{

int i ,j ,pos=0 ,min=0;

int swaps=0 ,comparisons=0;

for(i=0 ;i<n-1 ;i++)

{

pos=i;

for(j=i+1 ;j<n ;j++)

{

if(A[pos]>A[j])

{

// min=A[j];

pos=j;

}

comparisons++;

}

if(pos!=i)

{

int temp=A[pos];

A[pos]=A[i];

A[i]=temp;

}

swaps++;

}

printf("Sorted array :");

for(int i =0 ;i< n;i++)

{

printf("%d\t",A[i]);

}

printf("Total swaps %d and total comparisons %d",swaps ,comparisons);

}

int main()

{

int t ,n ,A[MAX];

printf("Enter the number of test cases :");

scanf("%d",&t);

while(t--)

{

printf("Enter the size of the array :");

scanf("%d",&n);

printf("Enter the array elements :");

for(int i=0 ;i<n ;i++)

{

scanf("%d",&A[i]);

}

selection\_sort(A,n);

}

return 0;

}

**OUTPUT**

**Enter the number of test cases :1**

**Enter the size of the array :6**

**Enter the array elements :12 89 32 56 0**

**1**

**Sorted array :0 1 12 32 56 89**

**Total swaps 5 and total comparisons 15**

1. **Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))**

**ANSWER III. PROGRAM**  
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#define MAX 100

void find\_duplicate(int A[] ,int lb ,int ub);

void merge(int A[] ,int lb ,int mid ,int ub);

void merge\_sort(int A[] ,int lb ,int ub);

void find\_duplicate(int A[] ,int lb ,int ub)

{

int flag=0;

while(lb<=ub)

{

int mid=(lb+ub)/2;

if( A[mid]==A[mid-1] || A[mid]==A[mid+1])

{printf("Duplicate element found");

flag=1;

break;}

else if((A[ub]+A[lb])/2==A[mid])

lb=mid+1;

else ub=mid-1;

}

if(flag==0)

{

printf("No duplicate element found");

}

}

void merge(int A[] ,int lb ,int mid ,int ub)

{

int n1=mid+1-lb;

int n2=ub-mid;

int L[MAX],R[MAX];

for(int i=0 ;i<n1 ;i++)

L[i]=A[i+lb];

for(int j=0; j<n2 ;j++)

R[j]=A[mid+1+j];

int i=0 ,j=0,k=0;

while(i<n1 && j<n2)

{

if(L[i]<R[i])

{

A[k]=L[i];

i++;

}

else{

A[k]=R[j];

j++;

}

k++;

}

while(i<n1)

{

A[k]=L[i];

k++;

i++;

}

while(j<n2)

{

A[k]=R[j];

j++;

k++;

}

}

void merge\_sort(int A[] ,int lb ,int ub)

{

if(lb>=ub)

{

int mid=(ub+lb)/2;

merge\_sort(A,lb,mid);

merge\_sort(A,mid+1 ,ub);

merge(A ,lb ,mid ,ub);

}

}

int main()

{

int result,result2;

int key;

int A[MAX],n;

int t;

scanf("%d",&t);

while(t--)

{

printf("\nEnter the size of the array :");

scanf("%d",&n);

printf("Enter the elements in the array :");

for(int i=0 ;i<n ;i++)

{

scanf("%d",&A[i]);

}

merge\_sort(A ,0 ,n-1);

find\_duplicate(A,0 ,n-1);

}

return 0;

}

**OUTPUT**

**1**

**Enter the size of the array :8**

**Enter the elements in the array :12 9 2 0 3 6 7 1**

**No duplicate element found**