O·ARAVIND
AP19110010523
CSE-H.

ASSIGNMENT-6

1 Ans:)

```
# include <stdio.h>
int main ( )
{
int i, low , high , mid , n , key , arr [100] , temp , i, one, two , sum , product;
printf ("Enter the number of elements in array ");
scanf ("%d", &n);
printf (" Enter %d integers ,"n);
for (i=0 ; i<n ; i++)
scanf ("%d", & arr[i] );
for (i=0 ; i<n ; i++)
{
  if (i=i+1 ; j<n ; j++)
  {
    if (arr [i] <arr[j])
    {
      if (temp = arr [i] );
      {
        arr [i] = arr [j] ;
        arr [i] = temp ;
      }
    }
  }
}
printf ("In elements of array is sorted in decending order : \n" );
for (i=0; i<n ; i++)
{
  printf ("%d", arr[i]);
}
printf (" Enter value to find");
scanf ("%d", & key);
low = 0·
high = n-1;
```

```
mid = (low + high) /2;
while (low < high)
{
   if (arr [mid ] < key)
   {
    low = mid + 1;
   }
   else if (arr [mid]= key )
   {
     printf ("%d found at location %d", key , mid +1);
     break;
  }
  else
   high = mid - 1;
   mid = (low + high /2 ;
}
 if ( low >high)
 {
  printf ("Not found %d isn't present in the list.n", key );
 }
 printf ("\n");
 printf ("Enter two locations to find sum and product of the elements");
 scanf ("%d", &one );
 scanf ("%d", &two);
  sum = (arr [one ]+ arr [two ]);
  product = (arr [one] * arr [two]);
  printf ("The sum of elements = %d", sum);
  printf (" The sum of elements = %d", sum ;product );
  return 0;
}
```

OUTPUT:-

```
Enter number of elements in array 5
Enter 5 integers
9
7
5
4
2
Element of array is sorted in descending order
97542 Enter value to find 5
 5 found at location 3
Enter two locations to find sum & product of the elements.
2
4
 The sum of elements = 87.
 The product of elements = 10.
```

2sol)

```
   # include <stdio.h>
   # include <conio.h>
   # define MAX_size 5.
  void merge_sort [MAX_size];
  void merge_array (int, int, int, int);
  int arr_sort [MAX_size];
  int main ( )
 {
  int i, k, pro = 1;
  printf ("Sample merge sort example functions & array \n");
 printf ("\n Enter %d Elements for sorting \n", MAX_size);
 for (i=0; i<max_size; i++)
 {
  scanf ("%d", & arr_sort [i]);
  printf ("In your data ");
 }
 for (i=0 , i<MAX_size ; i++)
  {
   printf ("\t%" , &arr_sort [i]);
  };
  merg_sort (0, MAX_size - 1);
  printf ("\n . sorted data :");
  for (i=0 , i<MAX_size ; i++)
  {
   printf ("\t%d ," arr_sort [i]);
 }
printf (" find the product of the kth element from first & last wher k \n");
scanf (" %d ", &k);
 Pro = arr_sort [k] * arr_sort [MAX_size - k - 1];
printf (" Produce = %d ", pro);
 getch ();
 }
 void merge_sort (int i, int j)
 {
   int m;
   if (i<j)
    {
     m= (i+j) /2;
     merge_sort (i,m);
     merge_sort (m+1, j);
   // merging two arrays
         merge_array (i, m, m+1, j);
     }
 }
void merge_array (int a , int b, int c , int d )
```

```
ind t[50];
inti=a ,j=c ,k=0;
while (i<b && j<=d)
{
  if (arr_sort[i] < arr_sort[j])
    t[k++ ]= arr_sort[i++];
  else
    t[k++ ] = arr_sort[j++];
}
/ collect remaining elements
while (i<=b)
      t[k+ +]= arr_sort [j++];
  for (i=a ,j=0, i<=d; i++ ;j++)
  arr_sort [i] = t[j];
}
```

OUTPUT:-

```
Sample merge sort example_functions and array:
Enter 5 elements for sorting
9
7
4
6
2
your data : 9 7 4 6 2
sorted data : 2 4 6 7 9
Find the product of kth elements from first & last k=2
Product =36.
```

3 Ans·) Insertion Sort:-

Insertion sort works by inserting the set of values in the existing sorted file. It constructs the sorted array by inserting a single element at a time. This process continues untill whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its appreciate place in the final list. The insertion sort method saves an effective ammount of memory.

Working of insertion sort:-

→ It uses two sets of arrays where one stores the sorted data & other on unsorted data.

→ The sorting algorithm works untill there are elements in the unsorted set.

→ Lets assume there are 'n' numbers elements in the array. Initially, the element with index 0 (LB = 0) exists in the sorted set remaining elements are in the unsorted partition of the list.

→ The first element of the unsorted position has array index 1 (if LB = 0)

→ After each iteration, it chooses the first element of the insorted position & inserts it into the proper place in the sorted set.

Advantages of Insertion sort:-

→ Easily implemented and very efficient when used with small sets of data

→ It is faster than other sorting algorithms.

Complexity of insertion sort:-

The best case complexity of insertion sort is O(n) times, i.e when the array is periously sorted. In the same way, when the array is sorted in the reverse order, the first element in the unsorted array is to be composed with each element in the unsorted array is to be compose.

with each element in the sorted set, i.e (O(n2)). In average case also it has to make the minimum (k-1)/2 comparisions. Hence the average case also has quadratic running time O(n²).

Example:-

arr [ ] = 46 22 11 20 9

// Find the minimum element in arr [0... 4] & place at begining

9 46 22 11 20

// Find the minimum element in arr [1... 4] & place at beging of arr [1... 4]

9 11 46 22 20

// Find the minimum element in arr [2... 4] & place at begining of arr[2...4]

9 11 20 46 22.

// Find the minimum element in the array a [3...4] & insert at the begining of the array [3... 4]

∴ Sorted array

9 11 20 22 46.

Slection sort:-

The selection sort perform sorting by searching for the minimum value number & placing it into the first or last position according to the order (ascending or decending). The process of searching the minimum key & placing it in the proper position at right position.

Working of the selection sort:-

→ Suppose an array Arr with n element in the memory.

→ In the second pass, again the position of the smallest value is determined in the sub array of (n-1) elements. interchange the Arr [pos] with Arr[i]

→ In the pass (n-1), the same process is performed to sort the n number of elements.

Advantages of selection sort:-

+ The main advantage of selection sort is that is performes well on a small list.

Complexity of selection sort:-

As the working of selection sort does not depend on the original order of the elements in the array so ther is not much different b/w best case & worst case complexity of selection sort. Similarly in the second pass also to find the second smallest element we given scanning of rest n-1 elements & the process is continued till the whole array sorted this scanning time complexity of selection sort is $O(n^2)$ =

$$(n-1)+(n-2)+ \dots + 2+1 = n(n-1)/2 - O(n^2)$$

Example:-

13 12 14 6 7

Let us loop for i=1 (second element of the array) to 4 (last element of the array)

i=1 . since 12 is smaller than 13, more 13 & insert 12 before 13.

do same for i=2, i=3, i=4

∴ Sorted array

6 7 12 13 14.

4sol)
```
# include <stdio.h>
# include <conio.h>
int main ( )
{
  int arr [50], i,j,n, temp, sum = 0, product = 1;
  printf (" Enter total number of elements to store ")
  scanf ("%d", &n);
  printf (" Enter %d elements ",n);
```

```
for (i=0; i<n, i++)
scanf ("%d", & arr[i]);
printf ("\n sorting array using bubble sort techique");
for (i=0 ; i<(n-1); i++ );
{
  for (j=0 ; j<(n-i-1); j++)
  {
    if arr [j] > arr[j+1]
    {
      temp = arr [j];
       arr [j] = arr [j+1]
       arr [j+1] = temp ;
    }
  }
printf (" All array elements sorted successfully \n");
printf (" Array elements in ascending order : \n\n");
for (i=0 , i<n , i++)
{
  printf (" %d \n", arr [i]);
}
printf (" array elements in alternate order \n" );
for (i=0 , i<=n ; i=i+2)
{
  printf (" %d \n", arr [i]);
}
for (i=1 ; i<=n ; i=i+2)
{
  sum = sum + arr [i];
}
printf (" the sum of odd position elements are = %d \n", sum );
for (i=0 ; i<=n ; i=i+2 )
```

```
  {
   product *=arr [i];
  }
 printf ("The products of even position elements are= %.d\n", product );
 getch ( );
 return 0 ( );
}
```

OUTPUT:-

```
 Enter total number of elements to store = 5.
 Enter 5 elements
  8
  6
  4
  3
  2
Sorted array using bubble sort technique
All array elements sorted successfully
Array elements in ascending order
 2
 3
 4
  6
 8
even elements in alternate order
2
4
8
 The sum of odd position element is 9
The product of even position element are 6,4.
```

```
5Ans:) # include <stdio.h>
  # include <stdio.h>
  void binary search ( int arr [ ], int num , int first ,int last )
  {
  int mid ;
    if ( first > last )
     {
      printf (" number is not found ");
     }
     else
      {
       mid = (first + last )/2 ;
      }
      if (arr [mid ]= =num )
      {
       printf (" Element is found at index %d ", mid );
       exit (0);
      }
      else if ( arr [mid ] >num )
      {
       primary search (arr , num , first , mid - 1);
      }
     else
      {
      Binary search (arr ; num , mid +1 , last );
    }
  }
}
void main ()
{
  int arr [100 ], beg , mid , end , i , n , num;
  printf (" Enter the size of an array ");
  scanf (" %d ", &n);
  printf ("Enter the value in sorted sequence \n");
```

```
for (i=0 ; i<n; i++)
{
  scanf ("%d", & arr[i]);
}
beg = 0;
end = n-1;
printf ("Enter a value to be search:");
scanf (" %d", & num)
 Binary search (arr , num, beg, end);
}
```

OUTPUT:-

```
Enter the size of an array 5
Enter the value in sorted sequence
4
5
6
7
8
Enter a value to search 5
Element is found at index: 1.
```