ASSIGNMENT - 6

M. Chandrahas
AP19110010536
CSE-F

3)

Insertion Sort: It works by inserting the set of values in the existing sorted file. It constructed the sorted array by inserting a single element at a time. The process continues until whole array is sorted in some order. The primary concept behind Insertion sort is to insert each item into its appropriate place in the final list. The insertion sort method save an efficient amount of memory.

Advantages of Insertion Sort:

* It is faster than other sorting algorithms
* The additional memory space requirement of insertion sort is less
* Easily implemented and very efficient when used with small lots of data.

Example:

25	15	30	9	99	20	26
15	25	30	9	99	20	26
15	25	30	9	99	20	26
9	15	25	30	99	20	26
9	15	25	30	99	20	26
9	15	20	25	30	99	26

Selection Sort:- It Performs Sorting by searching for the minimum value number and placing it into the first or left position according to the order. The Process of searching the min key and placing it in the proper position is continued until all the elements are placed at right position.

Advantages of Selection Sort:

* Simple to understand the sorting of elements in memory.
* It doesn't depend on the initial arrangement of elements.
* Suppose an array ARRAY with N elements in the memory.

Example:-

	0	1	2	3	4
1 →	17	16	3	15	6
	17 min	16	3 loc	15	6
2 →	3	16 min	17	15	6 loc
3 →	3	6	17 min	15 loc	6
4 →	3	6	15	17 min	16 loc
5 →	3	6	15	16	17

1)
```
#include<stdio.h>
int binary_search(int arr[], int a, int b, int x)
{
  if (b >= a){
    int mid = a + (b - a)/2;
    if (arr[mid] == x)
      return mid;
    if (arr[mid] > x)
      return binary_search(arr, a, mid-1, x);
    return binary_search(arr, mid+1, b, x);
  } return -1;
}
int main()
{
  int num;
  printf("Enter the size of array:");
  scanf("%d", &num);
  int i, j, a, val[num], op, var, p1, p2, sum, pro;
  for(a=0; a<num; a++)
  {
    printf("Enter value");
    scanf("%d", &val[a]);
  }
  for(i=0; i<num; i++)
  {
    for(j=i+1; j<num; j++)
    {
      if(val[i] < val[j])
      {
```

```
        a = val[i];
        val[i] = val[j];
        val[j] = a;
      }
    }
}
Print f(" It is in decreasing order:\n");
for(i=0; i<num; i++)
{
   Print f("%d", val[i]);
}
Print f("\n List \n");
Print f("1. Find val \n 2. Find Position \n 3. Printing ");
Print f("\n enter choice :\n");
Scan f("%d", &OP);
Switch (OP)
{
Case 1:
   Print f(" Enter Position")
   Scan f("%d", & var);

   break;

   Case 2:
   print f(" enter element ");
   Scan f("%d", & var);

   int result = binary search (var, 0, num-1, val);

   (result == -1) ? Print f("element not present");

   return 0;

   Case 3:
   Scan f("%d %d", &P1, &P2);
```

```
        sum = val[p1] + val[p2];
        PRO = val[p1] * val[p2];
        Print("SUM = %d\n", sum);
        break;
    }
}
```

2.)

```
#include <stdlib.h>
#include <stdio.h>
void merge(int arr[], int l, int m, int r
{
  int i, j, k;
  int n1 = m - l + 1;
  int n2 = r - m;
  int L[n1], R[n2];
  for(i = 0; i < n1; i++)
     L[i] = arr[l + i];
  for (j = 0; j < n2; j++)
     R[j] = arr[m + 1 + j];
  i = 0;
  j = 0;
  k = l;
  while (i < n1 && j < n2)
  {
     if( L[i] <= R[j])
     {
        arr[k] = L[i];
        i++
     }
     else
```

```
    }
      arr[k] = R[j];
      i++;
      k++;
    }
    while(j < n2)
    {
      arr[k] = R[j];
      j++;
      k++;
    }
}
void mergeSort(int arr[], int l, int r)
{
    if(l < r)
    {
      int m = l+(r-l)/2;
      mergeSort(arr, l, m);
      mergeSort(arr, m+1, r);
      merge(arr, l, m, r);
    }
}

void printArray(int A[], int size)
{
    int i;
    for(i=0; i < size; i++)
        printf("%d", A[i]);
    printf("\n");
}
```

```
}
int main()
{
    int size, v;
    Printf("enter size");
    Scanf("%d", &size);
    int val[size];
    for(v=0; v<size; v++)
    {
        Printf("enter value:");
        Scanf("%d", &val[v]);
    }
    Printf("Given val:");
    ~~Conf~~
    mergesort(val, 0, size-1);
    Printf("In sorted array is");
    int k, f, l, p1, p2, temp;
    Printf("enter value of k");
    Scanf("%d", &k);
    p1 = p2 = 1
    for(f=0; f<=k; f++)
    {
     Temp = val[f];
     p1 *= temp;
    }
    for(l=size-1; l>k; l--)
    {
    temp = val[l];
```

```
    *p2 = temp
  }
  printf(" : %d %d", p1, p2);
}
```

4)
```
#include <stdio.h>
void bubble sort(int ar[], int n)
{
  int i, j, temp;
  for(i=0; i<n-1; i++)
  for(j=0; j<n-i-1; j++)
  if(ar[j] > ar[j+1])
  {
    temp = ar[j];
    ar[j] = ar[j+1];
    ar[j+1] = temp
  }
}
int main()
{
  int siz, i;
  printf("enter size");
  scanf("%d", &siz);
  int arr[siz]
  for(i=0, i<siz; i++)
  {
    printf("enter element");
    scanf("%d", &arr[i]);
  }
```

```
bubble sort(arr, len);
Printf("sorted array:");
for(i=0; i<len; i++)
{
    Printf("%d", arr[i]);
    Printf("(#");
}
Printf("menu");
Printf("1. display");
Printf("2. sum of evens in 3. divisible by m\n");
int op, sum=0, product=1, m;
Printf("enter choice");
Scanf("%d", &op);
switch(op)
{
    case 1:
    for(i=0; i<len; i+=2)
    {
        Printf("%d\t", arr[i]);
    }
    case 2:
    for(i=0; i<len; i+=2)
    {
        sum = sum + arr[i];
    }
    for(i=1; i<len; i+=2)
    {
        product = product * arr[i];
    }
    Printf("sum: %d\n", sum);
    Printf("product: %d\n", product);
```

```
        cole3:
        PrintA("enter value m:")
        ScanA("%d",&m);
        PrintA("divisible numbers %d are:",m);
        for(i=0;i<lis;i++)
        {
          iA(ars[i]%m==0)
          {
            PrintA("%d\d",ars[i]);
          }
        }
    }
}
```

5)
```
// include <stdio.h>
int binarySearch(int a[], int c, int m, int x)
{
    int mid=(c+m)/2;
    iA(c>m) return -1
    if(a[mid]==x)
    return mid;
    iA(a[mid]<x)
    return binarySearch(a,mid+1,m,x);
    else
    return binarySearch(a,c,mid-1,x);
}
int main(void)
{
```

```
int a[100];
int len, Pos, val;
Print f("enter length ");
Scan f("%d", & len);
Print f(" enter elements");
for(int i=0; i<len; i++)
Scan f("%d", & a[i]);
print f("enter element to search");
Scan f("%d", & val);
Pos = binary search (a, 0, len-1, val);
if (Pos <0)
    printf(" %d", val);
  else
    Print f(" %d, %d ", val, Pos+1);
return;
}
```