DSA Assignment-6

G. Sai Sushma,
AP19110010551,
CSE-G.

① Sol:

code:

```
#include <stdio.h>
#define NUM 30
void bubblesort (int array[], int size)
{
 for(int i=0; i<size-1; ++i)
  for (int j=0; j<size-i-1; ++j)
  {
   if(array[j] < array[j+1])
    {
     int temp = array[j];
     array[j] = array[j+1];
     array[j+1] = temp;
     }
   }
}
 void display (int array[], int size)
 {
 for (int i=0; i<size; ++i)
 { printf ("%d", array[i]);
  }
  printf ("\n");
  }
 int binary search (int array [], int l, int r, int x)
 {
  if (r>=l)
  {
   int mid= l+ (r-l)/2;
   if (array[mid]) == x)
  {
```

```
return mid;
}
else if (array[mid] > x)
{
 return binary search(array, l, mid -1 , x);
 }
 else
 {
  return binary search (array , mid +1, r, x);
  }
 }
 return -1;
}
   void sum and product (int array[])
   {
     int loc1, loc2;
     printf ("Enter location 1: ");
     scanf("%d", & loc1);
     printf ("Enter location 2: ");
     scanf ("%d", & loc2);
     printf ("sum of elements in positions %d and %d is; %d\n",
            loc1, loc2, array [loc1 -1 ]+ [array[loc2 -1]];
      printf (" product of elements in positions %d and %d is;
      %d\n", loc1, loc2, array [loc1-1]* array [loc 2-1]);
     }
     int main()
   {
     int a [NUM], size, k, r, result;
     print ("Enter no. of elements of array: ");
     scanf ("%d", & size);
     for (k=0; k<size; k++)
```

```
{
  printf("Enter the %dth element:", k+1);
  scanf("%d", &a[k]);
}
  printf("Given array: \n");
  display (a, size);
  bubble sort (a, size);
  printf(" Sorted Array in Descending Order: \n");
  display (a, size);
  printf (("a) \n");

  printf ("Enter the element to search:");
  scanf("%d", &r);
  result = binary search (a, 0, size-1, r);
  if (result == -1)
  {
    printf("%d element is not found in sorted array\n", r); }
  }
  else
  {
    printf(" %d element is found in sorted array at location %d\n",
           r, result+1);

  }
     printf ("b) \n");
     sum and product(a);
     return 0;
  }
```

<u>output</u>:

```
Enter no. of elements of array : 3

 Enter the 1th element :12
 Enter the 2th element : 56
 Enter the 3th element : 78
```

Given array:

12. 56 78

sorted Array in Descending order:

~~78~~ 78 56 12

a)

```
Enter the element to search : 12
12 element is not found in sorted array.
```

b)

```
Enter location 1:12
Enter location 2:56
sum & elements in positions 12 and 56 is, : 326800
product & elements in positions 12 and 56 is : 0
```

2) Sol: code:

```
#include <stdio.h>
#define ms 100
int a[ms];
void merge (int l1, int u1, int l2, int u2)
{
    int i,j,k, temp [ms];
        k=0;
        i=l1;
        j=l2;
        while ((i<=u1) && (j<=u2))
        {
         if (a[i] < a[j])
         {
          temp[k] = a[i]; i++; k++
         }
else
{
 temp[k] = a[j]; j++; k++;
 }
}
 while (i<=u1)
 {
  temp[k] = a[i]; i++, k++;
 }
 while (j<=u2)
 {
  temp[k] = a[j] ; j++; k++;
  }
 for (i=l1, K=0; i<=u2; i++, k++)
  {
   a[i] = temp[k];
   }
 }
void mergesort(int lb, int ub)
{
 if (lb<ub)
 {
   int mid = (ub+lb)/2;
   mergesort (lb,mid);
   merge sort (mid+1,ub);
   merge (lb,mid,mid+1,ub);
   }
}
 int main()
 {int i,n, product =1, K;
```

```
printf("\n Enter the size of the array max (100) ");
scanf ("%d", &n);
for (i=0; i<n; i++)
{
   printf("a [%d]\t = ", i);
   scanf ("%d", &a [i]);
}
 mergesort (0,n-1);
 printf ("Enter k\n");
 scanf("%d", &k);
 for (i=0; i<k; i++)
 { product *= a[i];
 }
  printf("\n The product till the kth element is %d\n", product);
  return 0;
}
```

Out put:

```
Enter the size of the array max(100) : 5
  a [0]    = 3
  a [1]    = 5
  a [2]    = 6
  a (3)    = 1
  a [4]    = 8
  Enter k
  The product till the kth element is
  90.
```

3) Insertion sort:

Definition:

Insertion sort works by inserting the set of values in the existing sorted site. It can be constructed. the sorted array by inserting a single element at a time. This process continous until whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its ~~approximation~~ appropiate place in the final list. The insertion sort method save an effective amount of memory.

Advantages of Insertion sort:

* Easily implemented and very efficient when used with small sets of data.
* The additional memory space Requirement of Insertion sort is less (i.e o(1))
* It is consider to be live sorting techniques as the list can be sorted as the new elements are received.
* It is faster than other sorting algorithems.

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
9	15	20	25	26	30	99

Selection sort:

Definition:

The selection sort perform sorting by searching for the minimum value number and placing it into the first or last position according to the order. The process of searching the minimum key and placing it in the proper position is continoued untill the all the elements are placed at right position.

Advantages of selection sort:

* Suppose an array ARR. with N elements inthe memory.
* Simple to understand the sorting of elements doesn't depend on the initial arrangment of the elements.

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	16
4 →	3	6	15	17 min	16 Loc
5 →	3	6	15	16	17.

4
Sol:

code:

```
# include<stdio.h>
# define NUM 30
void bubblesort (int array[], int size)
{
 for (int i=0; i<size-1; ++i)
  for (int j=0; j<size-i-1; ++j)
   {
    if (array [j] > array [j+1])
    {
     int temp = array [j];
     array [j] = array [j+1];
     array [j+1] = temp;
    }
   }
}
  void display (int array[], int size)
  { for (int i=0; i<size; ++i)
    { printf ("%d", array(i]);
    }
    printf("\n");
  }
   void alternate (intarray[], int size)
   { for (int i=0; i<size; i= i+2)
     { print-f ("%d", array[i]);
      }
      printf ("\n");
    }
```

```
void sumandproduct (int array[], int size)
{
  int sum=0, product =1;
  for (int i=0; i<size; i=i+2)
  {
    sum = sum+array[i];
  }
  for (int j=1; j<size ; j=j+2)
  {
   product = product*array[j];
  }
   printf ("sum of elements in odd position: %d\n", sum);
   printf(" product of elements in even position : %d\n", product);
 }
 void divisible (int array[], int size)
 {
    int m;
    printf("Enter the value of m:");
    scanf ("%d", &m);
    printf ("Elements of array divisible by %d are:\n", m);
     for(int i=0; i<size; i++)
     {
       if(array [i] %m ==0)
        {
         printf ("%d", array[i]);
        }
     }
 }
int main()
{
```

```
int a[NUM], size,k;
printf ("Enter no. of elements of array:");
scanf ("%d", &size);
for (k=0; k<size; k++)
{
  printf ("Enter the %dth element : ", k+1);
  scanf ("%d", &a[k]);
}
 printf ("Given array : \n");
 display (a, size);
 bubblesort ( a, size);
 printf ("sorted Array in Ascending Order : \n");
  display (a, size);
  printf ("a)\n");
 printf ("sorted Array in Alternate order.\n");
 alternate (a, size);
  printf ("b)\n");
  sum and product (a, size);
  printf ("c)\n");
 divisible (a, size);
 return 0;
}
```

out put:

```
Enter no. of elements of array : 4
 Enter 1th element : 60
 Enter 2th element : 20
 Enter 3th element : 40
 Enter 4th element : 80
```

```
Given array :
 60  20  40 80
Sorted array in Ascending order:
 20  40   6080
a)
   sorted Array in Alternate order:
    20   60
b) sum of elements in odd position : 80
   product of elements in even position: 3200
 c)
   Enter the value of m: 2
   Elements of array divisible by 2 are :
      20    40    60  80
```

5)
sol:

code :

```
#include <stdio.h>
void binary_search(int[], int,int,int);
 void bubble_sort  (int[], int);

 int main()
{
   int Key, size, i;
   int list [25];
  printf ("Enter size of a list : ");
  scanf ("%d", &size);
  printf (" Enter elements\n ");
  for (i=0; i<size ; i++)
  {
     scanf ("%d", & list [i]);
  }
```

```
    bubble_sort (list, size);
    printf ("\n ");
    printf ("Enter key to search\n");
    scanf ("%d", & key);
     binary_search (list, 0, size, key);
}
   void bubble_sort (int list[], int size)
    {
      int temp, i, j;
      for (i=0; i<size; i++)
      {  for (j=1; j<size ; j++)
          {
           if (list[i] > list[j])
           {
             temp = list[i];
             list[i] = list [j];
             list [j] = temp;
           }
         }
       }
}
     void binary - search (int list[], int lo, int hi, int key)
    {  int mid;
       if (lo >hi)
       {
         printf ("key not found\n");
         return ;
       }
        mid = (lo+hi) /2;
        if (list[mid] == key)
```

```
{
 printf ("key found\n");
}
else if (list (mid) > key)
{
  binary_search(list,10, mid -1, key);
}
 else if (list (mid] < key)
 {
   binary_search (list, mid+1, hi, key);
 }
}
```

out put:

```
    Enter the size of the list : 4
     Enter elements
      3
      4
      5
      6
     Enter Key to search
     4
     Key found.
```

* THE END *