DSA- Assignment-6

D. Hasika
AP19110010499
CSE-G.

1. Take the elements from the user and sos them in the descending order and do the following.
a. using binary search find the elements and the location in the elements is askes from user.
b. Ask the user to enter any TWO locations print the sum and product of values at those locations in the sosted array.

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

```
    return binary seasch (assay, l, mid -1, x);
    }
    else {
      return binary seasch (assay, mid+1, r, x);
    }
    }
    return -1;
    }
    void sum and product (int assay[]){
      int loc1, loc2;
      printf ("entes location 1:");
      scanf ("%d", &loc 1);
      printf ("entes location 2:");
      scanf ("%d", &loc 2);
printf ("sum of elements in positions %d & %d is %d \n",
  loc 1, loc 2, assay [loc1-1]+assay [loc 2-1]);
printf ("product of elements in postions %d & %d dis: %d
\n", loc1, loc2, assay [loc 1 -1] *assay [loc2-1]); }
int main( )
{
int a[NUM], size, k, r, result;
printf ("entes no. of elements of assay:");
scanf ("%d", &size);
for (k=0; k<size; k++)
{
printf ("entes the %dth element: ", k+1);
scanf ("%d", &a[k]);
}
printf ("given assay: \n");
display (a, size);
bubble sort (a, size);
printf ("sorted Array in descending order \n");
display (a, size);
printf ("entes the element to seasch: ");
scanf ("%d", &r);
result = binary seasch (a, 0, size-1, r);
```

```
if (result == -1)
{
printf ("%d element is found in sorted
        assay in ", r);
  }
  else {
  printf("%d element is found in sorted
assay at location %d in", r, result+1);
  }
  printf ("\n");
  Sum and product (a);
  return 0;
  }.
```

2. Sort the array using merge sort where elements are taken from the user and find the product of k^{th} elements from first and last where k is taken from the user.

```
#include <stdio.h>
using namespace std;
#define n 4
// merge arr1[0..n1-1] and arr2[0..n2-1] into
// arr3[0..n1+n2-1]
void mergeArrays(int arr1[], int arr2[], int n1,
                 int n2, int arr3[])
{
  int i=0, j=0, k=0;
  while (i<n1 && j<n2)
  {
    if (arr1[i] < arr2[j])
      arr3[k++] = arr1[i++];
    else
      arr3[k++] = arr2[j++];
  }
  while(i<n1)
    arr3[k++] = arr1[i++];
  while (j<n2)
    arr3[k++] = arr2[j++];
  void printArray(int arr[], int size)
  {
    for(int i=0; i<size; i++)
      cout<< arr[i] << " ";
  void merge k arrays(int arr[][n], int i, int j,
                      int output[])
  {
    int
```

```
if (i==j)
{
for (int p=0; p<n; p++)
   [p] = arr[1] [p];
   return;
}
if (j-i==1)
{
merge Arrays (arr[i], arr[j], n, n.);
  return;
}.
```

3. Discuss Insertion sort and selection sort with examples?

Insertion sort

Definition: Insertion sort 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. This process continues until whole assay is sorted in same 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 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 considered to be live sorting techniques as the list can be sorted as the new elements are received
* It is faster than other sorting algorithms.

Example of Insertion sort

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 perfom sorting by searching for the min value numbes and placing it into the first of last position according to the order. The process of searching the minimum key and placing it in the proper position is continuced until the all the elements are placed at right position.

Advantages of selection sort

* Suppose an assay ARR. with N elements in the mm memory.
* simple to understand the sorting of elements doesn't depend on the initial assangment of the elements.
* The selection sort carries out the selection and positioning of the required elements.

Example of selection sort

	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. sort the array using bubble sort where elements are taken from the user and display the elements
   i, in alternate order.
   ii, sum of elements in odd position product in even
   iii, elements divisible by m.

```
#include<stdio.h>
#define NUM 30
void bubble sort (int assay[], int size)
{
 for (int i=0; i<size-1; i++)
  for(int j=0; j<size-i-1; j++)
  {
   if (assay[j] > assay[j+1]
   {
    int temp = assay[j];
    assay[j] = assay[j+1];
    assay[j+1] = temp;
   }
  }
}
void display (int assay[], int size)
{
```

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

```
    for (int i=0; i<size; i++){
        if (assay[i]%m==0){
            printf("%d", assay[i]);
        }
    }
}
int main()
{
    int a[NUM], size, k;
printf("Enter no. of elements of assay:");
scanf("%d", &size);
for (k=0; k<size; k++)
{
printf("Enter the %dth element :", k+1);
scanf("%d", &a[k]);
}
printf("Given assay: \n");
display(a, size);
bubblesort(a, size);
printf("Sorted Array in Asending Order \n");
display(a, size);
printf("\n\n");
printf("Sorted Array in Alternet Order: \n");
alternate(a, size);
printf("b)\n");
sum and product(a, size);
printf("c)\n");
divisible(a, size);
return 0;
}
```

5. write a Recursive program to implement binary search?

```
#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 ");
  for (i=0; i< size; i++)
  {
   scanf ("%d", &list(i));
  }
  bubble.
  binary search
  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=i; 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, lo, mid - 1, key);
 }
 else if (list[mid] < key)
 {
 binary-search(list, mid+1, hi, key);
 }
}.
```

*** THE END ***