

DSA-Assignment-6

D. Hasika
API9110010499
CSE-G.

1. Take the elements from the user and sort them in the descending order and do the following.
 - a. using binary search find the elements and the location in the elements is asked from user.
 - b. Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array.

```
#include <stdio.h>
```

```
#define NUM 30
```

```
void bubble sort (int array[], int size)
```

```
{  
    for (int i=0; i<size-1; i++)  
        for (int j=0; j<size-1-i; j++)
```

```
{  
    if (array[j] < array[j+1])  
        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 (l >= r) {
```

```
        int mid = l + (r-l)/2;  
        if (array[mid] == x) {  
            return mid;  
        }
```

```
        else if (array[mid] > x) {
```

```

return binary search (array, 1, mid - 1, x);
}
else {
    return binary search (array, mid + 1, r, x);
}
}
return -1;
}

void sum and product (int array[]) {
    int loc 1, loc 2;
    printf ("Enter location 1:");
    scanf ("%d", &loc 1);
    printf ("Enter location 2:");
    scanf ("%d", &loc 2);
    printf ("Sum of elements in positions %d & %d is %d\n",
        loc 1, loc 2, array [loc 1 - 1] + array [loc 2 - 1]);
    printf ("Product of elements in positions %d & %d is %d\n",
        loc 1, loc 2, array [loc 1 - 1] * array [loc 2 - 1]);
}

int main ()
{
    int a [NUM], size, k, r, result;
    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);
    bubble sort (a, size);
    printf ("Sorted Array in descending order:\n");
    display (a, size);
    printf ("Enter the element to search:");
    scanf ("%d", &r);
    result = binary search (a, 0, size - 1, r);
}

```

```

if (result == -1)
{
printf ("%d element is found in sorted
        array in ", r);
}
else {
printf ("%d element is found in sorted
array at location %d in ", r, result+1);
}
printf ("b\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 N4
```

```
// 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 mergeKarrays (int arr[][N], int i, int j,  
                  int output[])
```

```
{  
    for
```



```

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 constructs the sorted array by inserting a single element at a time. This process continues until the whole array 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 saves 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 a live sorting technique 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 perform sorting by searching for the min 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 continued until all the elements are placed at right position.

Advantages of Selection sort

- * Suppose an array ARR. with N elements in the memory.

- * simple to understand the sorting of elements doesn't depend on the initial arrangement 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	16	3	15	6
	min		loc		
2 →	3	16	17	15	6
		min		loc	
3 →	3	6	17	15	16
			min	loc	
4 →	3	6	15	17	16
				min	loc
5 →	3	6	15	16	17

4. sort the array using bubble sort whose 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 divide by m.

```
#include <stdio.h>
#define NUM 30
```

```
void bubble sort (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 (int array[], int size)
{

```

```

    for (int i=0; i<size; i=i+2) {
        printf ("%d", array[i]);
    }
    printf ("\n");
}

```

```

void sumand product (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:");
    display(a, size);
    bubbleSort(a, size);
    printf("Sorted Array in Ascending order:");
    display(a, size);
    printf("a)\n");
    printf("Sorted Array in Alternate order:");
    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-sort
```

```
    binary-search
```

```
    printf ("in")
```

```
    printf ("enter key to search in");
```

```
    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 ***