Y. Divya Sri
AP19110010036
CSE-G

Assignment-6

1. Take the elements from the user and sort them in descending order and do the following.

a) using Binary search find the element and the location in the array where the element is asked from the user.

b) Ask the user to enter any two locations point the sum and product of values at those locations in the sorted way.

code:

```
# include <stdio.h>
void sort (int a[], int n)
{
    int i, j, temp;
    for (i=0; i<n; i++)
    {
        for (j=i+1; j<n; j++)
        {
            if (a[i]<a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}
int binary (int a[], int c, int n)
{
    int i=0, j=n-1, mid;
    while (i<=j)
    {
        mid = (i+j)/2;
        if (a[mid] == c)
            return mid+1;
        else
        {
            if (c<a[mid])
                j = mid-1;
```

```
            else
                i = mid+1;
        }
    }
    if(i>j)
    {
      return 0;
    }
}
int main()
{
    int n,i,a[100], g, c, s1, s2;
    printf("enter the no. of elements of array");
    scanf("%d", &n);
    printf("enter the elements of array \n");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    sort(a, n);
    for(i=0; i<n; i++)
        printf("%d", a[i]);
    printf("enter the element to find in array");
    scanf("%d", &c);
    g = binary(a, c, n);
    if(g!=0)
    {
        printf("element is found at %d position", g);
    }
    else
    {
        printf("element not found\n");
    }
    printf("enter the position of array to find sum and
                                        product \n");
    scanf("%d %d %d", &s1, &s2);
```

③

```
s1--;
s2--;
printf("The sum is %d", a[s1]+a[s2]);
printf("The product is %d", a[s1]*a[s2]);
}
```

output:

```
Enter the no of elements of array 5
enter the elements of array
25
14
20
5
1
25 20 14 5 1 .
enter the element to find in array 15
element not found
enter the position of array to find sum and product
2
5
the sum is 21 the product is 20.
```

② Sort the array using merg sort where elements are taken from the user and find the product of kth elements from 1st and last where k is taken from the user.

coder

```
# include <stdio.h>
# include <stdlib.h>
// Merges two subarrays of arr[]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
        n2 = r - m;
    /* create temp arrays */
    int L[n1], R[n2];
```

(a)

```
/* copy data to temp arrays L[] and R[]*/
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; // initial index of 1st subarray.
   j=0; // initial index of 2nd subarray
  k=l; // initial index of merged subarray
  while (i<n1 && j<n2)
  {
     if (L[i] <= R[j])
     {
        arr[k] = L[i];
        i++;
      }
      else
       {
         arr[k] =R[j];
         j++;
       }
        k++;
   }
   /* copy the remaining elements of L[], if there are any */
   while (i<n1)
  {
   arr[k] = L[i]
   i++;
   k++;
}
/* copy the remaining elements of R[], if there are any*/
while (j<n2)
 {
    arr[k] = R[j];
```

```
        j++;
        k++;
      }
    }

/* l is for left index and r is right index of the sub-array of arr
 to be sorted */
void mergeSort(int arr[], int l, int r)
{
  if (l < r)
  {
  // Same as (l+r)/2, but avoids overflow for
  // large l and h
  int m = l+(r-l)/2;
  // Sort first and second halves
  mergeSort(arr, l, m);
  mergeSort(arr, m+1, r);
  merge(arr, l, m, r);
  }
}

/* Function to print an array */
void printArray(int A[], int size)
{
  int i;
  for (i=0; i<size; i++)
  printf("%d ", A[i]);
  printf("\n");
}

/* Driver program to test above functions */
int main()
{
  int arr[5];
  int i;
  int arr_size = sizeof(arr)/sizeof(arr[0]);
```

```
for (i=0; i<arr_size; i++) {
    printf("enter the elements");
    scanf("%d", &arr[i]);
}
printf("Given array is \n");
printArray(arr, arr_size);
mergeSort(arr, 0, arr_size-1);
printf("\n sorted array is \n");
printArray(arr, arr_size);
int k;
printf("enter the value of k");
scanf("%d", &k);
int fromfirst = arr[k-1];
int fromlast = arr[5-(k)];
printf("%d", fromlast * fromfirst);
return 0;
}
```

3) Discuss insertion sort and selection with examples.

Ans: **Selection Sort:**

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

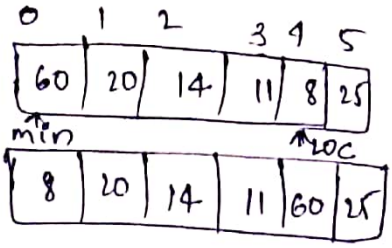
Following example explains the above steps:

arr[] = 60 20 14 11 8 25

// Find the minimum element in arr[0....4]

// and place it at beginning

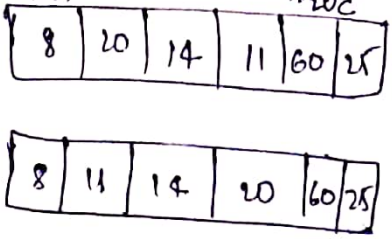
8 20 14 11 60 25



// Find the minimum element in arr[1...4]

// and place it at beginning of arr[1...4]

8 11 14 20 60 25



// Find the minimum element in arr[2...4]

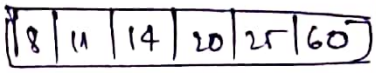
// and place it at beginning of arr[2...4]

8 11 14 20 60 25

// Find the minimum element in arr[3...4]

// and place it at beginning of arr[3...4]

8 11 14 20 25 60



Insertion Sort:

Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands.

Algorithm.

// sort an arr[] of size n

insertionSort(arr, n)

loop from i = 1 to n-1

.... a) pick element arr[i] and insert it into sorted sequence arr[0...i-1]

Example

89 17 8 12 0

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

i = 1. since 89 is greater than 17, move 89 and insert 17 before 89.

17 is smaller than 89

17 89 8 12 0

i=2. 8 is smaller than 89 and 17. move 17 and 89 and insert 8 before them (17 & 89).

8 17 89 12 0

i=3. 12 is smaller than 17 and 89. move 17 and 89 and insert 12 before them (17 & 89).

8 12 17 89 0

i=4. 0 is smaller than 89 17 12 8. move 89, 17, 12 and 8 and insert 0 before them (8, 12, 17, 89).

∴ The insertion solted order = 0 8 12 17 89.

4) Solt the array using bubble solt where elements are taken from the user. and display the elements.

i) In alternate order.

ii) sum of elements in odd positions & product of elements in even positions.

iii) Elements which are divisible by m where m is taken from the user.

Code

```
# include <stdio.h>
void main()
{
int a[100], n, i, j, temp, s=0, p=1, m;
Printf("enter no of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for(i=0; i<n; i++)
{
  scanf("%d", &a[i]);
}
for(i=0; i<n-1; i++)
{
  for(j=0; j<n-i-1; j++)
  {
```

```
    if (a[j] > a[j+1])
    {
      temp = a[j];
      a[j] = a[j+1]
      a[j+1] = temp;
    }
  }
}
printf("\n Sorted list in ascending order:\n");
for (i=0; i<n; i++)
{
 printf("%d\n", a[i]);
}
printf("The alternate order is:");
for (i=0; i<n; i++)
{
  if(i%2 == 0)
  {
    printf("%d", a[i]);
  }
}
for (i=0; i<n; i++)
{
  if (i%2 != 0)
  {
    s = s + a[i];
  }
}
printf("\n sum of odd index is %d", s);
for (i=0; i<n; i++)
{
  if (i%2 == 0)
  {
    p = p * a[i];
  }
}
```

```
printf("\n product of odd index is %d ", p);
printf("\n Enter the value of m\n");
scanf("%d", &m);
for (i=0; i<n; i++)
{
  if (a[i] %m ==0)
  {
    printf("%d", a[i]);
  }
}
}
```

output:-

```
Enter number of elements : 5
Enter 5 integers : 12 10 3 52 20
sorted list in ascending order : 3 10 12 20 52
the alternate order is 3 12 52
sum of odd index is 30
product of odd index is 1872
enter the value of m : 2
10 12 20 52
```

5) write a recursive program to implement binary search.

coder

```
# include <stdio.h>
int recursive Binary Search (int array[], int start_index,
                             int end_index, int element){
  if (end_index >= start_index){
    int middle = start_index + (end_index - start_index)/2;
    if (array[middle] == element)
      return middle;
    if (array[middle > element)
    return recursive Binary Search (array, start_index,
                                    middle-1, element);
  return recursive Binary Search (array, middle+1, end_index,
                                  element);
}
```

```
    return -1;
}
int main(void) {
    int array[] = { 3, 9, 12, 15, 27 };
    int n=5;
    int element = 15;
    int found_index = recursiveBinarySearch(array, 0, n-1, element);
    if (found_index == -1) {
        printf("element not found in the array");
    }
    else {
        printf("Element found at index : %d", found_index);
    }
    return 0;
}
```

output:

Element found at index 3.