DSA Assignment-6

B. HARITHA
CSE-G
AP19110010287

SEARCHING AND SORTING

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

a) Using binary search find the element the location in the array where the element 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 <stdlib.h>
# include <stdio.h>
int comparator (const void *p1, const void *p2) {
    return (*(int*)p2 - *(int*)p1);
}
int binarysearch (int arr[], int size, int search) {
  int beg = 0, end = size-1, mid;
  while (beg <= end) {
     mid = (beg + start) / 2;
     if (arr[mid] == search) {
         return mid;
     }
     else if (arr[mid] > search) {
         end = mid - 1;
     else beg = mid + 1;
  }
```

```
    return -1;
}
int main() {
    int arr[100], size, search, i, pos = -1, loc1, loc2;
    Printf("\n Enter the size of the array max (100)");
    scanf("%d", &size);
    Printf("\n Enter elements in array \n");
    for (i=0; i<size; i++) {
        scanf("%d", &arr[i]);
    }
qsort (arr, size, sizeof(int), comparator);
Printf("\n the sorted array is : \n");
for (i=0; i<size; i++)
Printf("%d", arr[i]);
Printf("\n Enter search elements");
Scanf("%d", &search);
Pos = binary search (arr, size, search);
if (pos == -1)
{ Printf("Not found \n");
}
else {
    Printf("\n the %d search element is found at index
            %d \n", Search, pos);
```

```
    Printf (" Enter two indexes \n");
    Scanf (" %d %d ", &loc1, &loc2);
    Printf (" Sum is %d \n", arr[loc1] + arr [loc2]);
    Printf (" Product is %d \n", arr [loc1] * arr [loc2]);
    }
}
```

Output :-

Enter the size of array elements (max (100)) 5

Enter the elements in array.

9 2 4 1 5

The sorted array is.

9 5 4 2 1

Enter the search element 5

The 5 search element is found at index 2

Enter two indexes

1 2

Sum is 9

Product is 20

2) Sort the array using Merge sort where elements are taken from the user and find the product of Kth elements from first and last where k is taken from the user.

```
# 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);
        mergesort (mid+1, ub);
        mergesort (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 k* element is %d \n", product);
    return 0;
}
```

Output :

Enter the size of the array 3

a[0] = 2

a[1] = 1

a[2] = 4

Enter the k

1

The product till the k-th element is 1

3) Discuss insertion sort and selection sort with examples.

Insertion Sort :-

Suppose an array A with n elements A[1], A[2], ... A[N] is in memory. The insertion sort algorithm scans A from A[1] to A[N], inserting each element A[K] into its proper position in the previous sorted sub array A[1], A[2] ... A[K-1].

Example :-

array initial : 5, 9, 2, 1, 8.

pass 1 : 5, 9, 2, 11, 8.

pass 2 : 2, 5, 9, 11, 8

pass 3 : 2, 5, 9, 11, 8

pass 4 : 2 5, 8, 9, 11 sorted.

Sudo code :

1. A[10] = minimum integer value.
2. Repeat steps 3 through 8 for k = 1, 2, 3 ... , N-1
3. {
   temp = A[k]
4. ptr = k-1
5. Repeat steps 6 to 7 while temp < A[ptr]
   {
6. A[ptr+1] = A[ptr]
7. ptr = ptr -1
   }
8. A[ptr+1] = temp
   }
9. End.

Time Complexity :

best : O(n) average $O(n^2/n)$ worst $O(n^2)$

Space complexity : O(1)

Selection Sort :-

The basic idea of selection sort is repeatedly select the smallest key in the unsorted array.

Example : 18, 6, 15, 4, (2) → smallest

Pass 1. 2 18, 6, 15, (4) → smallest

Pass 2. 2, 4, 18, 6, 15 → smallest

Pass 3 2, 4, 6 18, 15 → smallest

Pass 4 2, 4, 6, 15 18 → smallest

Pass 5 2, 4, 6, 15, 18 → sorted

Pseudocode:

```
1. Small = AR[L]
2. For l = 2 to U do {
3. small = AR[i), Pos = 1
4. for 5 = 1 to u do {
5.    if AR[j] < small {
        Small = AR[i], Pos = [·]
6. }
7. J = J + 1
8. temp = AR[i), AR[i] = small, AR[Pos] = temp.
   }
9. End
```

Time Complexity

best: O(n) average : O(n²) worst : O(n²)

Space Complexity O(1).

4.) Sort the array using bubble sort where elements are taken from user and display the elements.

i) in alternate order.

ii) sum of elements in odd positions and product of elements is even positions.

iii) Element which are divisible by m, where m is taken from user.

```
#include <stdio.h>
Void displayAltsum (int arr[], int size ) {
int i, sum =0, Product =1 ;
Printf (" Alternate elements \n") ;
for ( i =0 ; i< size ; i++) {
 if ( i % 2 != 0 ) {
      Product += arr[i];
   }
  else {
       sum + = arr [i];
       Printf (" %d ", arr [i] );
      }
 }
 Printf ("\n Sum of the odd elements = %d \n", sum);
 Printf (" \n product of the even elements = %d \n", Product);
}
```

```
void divM(int arr[], int size) {
    int i=0, m;
    Printf("enter the m\n");
    Scanf("%d", &m);
    Printf("elements divisible by %d \n", m);
    for (i=0; i<size; i++) {
        if (arr[i] %m == 0)
            printf("%d", arr[i]);
    }
}
Void bubble sort(int arr[], int size)
{
    int i, j, temp;
    for (i=0; j<size-1; i++)
        for (j=0; j<size-i-1; j++)
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
    displayAltSumpro (arr, size);
        divm (arr, size);
    }
int main()
{
    int arr[100], size; i;
```

```
Printf ("\n Enter the size of the array (max 100) ");
Scanf ("%d", &size);
Printf ("\n Enter elements in array \n");
for (i=0; i<size; i++){
    scanf ("%d", &arr[i]);
}
bubblesort (arr, size-1);
return 0;
}
```

Output :-

Enter the size of the array (max 100) 5
Enter the elements in array.
8
3
5
1
7
Alternate elements.
1 5
Sum of the add elements = 6
Product of the even elements = 6
Enter the m
2
elements divisible by 2
8

5.) write a recursive program to implement binary search?

```
# include <stdio.h>
int binary search (int arr[], int beg, int end, int search) {
 int mid ;
if (beg <= end) {
 mid = (beg + end) /2 ;
if (arr [mid] == search ) return mid ;
if (arr [mid] > search)
    return binary search (arr, beg, mid -1, search);
  return binary search (arr, mid +1 , end, search) ;
}
 return -1 ;
}
int main () {
   int arr [100], size, search, i, pos;
   Printf ("\n Enter the size of the array (max 100) ");
   Scanf ("%d ", & size);
   Printf ("%d ", & \n Enter sorted elements in array \n");
   for (i=0 ; i<size ; i++) {
       Scanf ("%d , &arr[i]) ;
  }
   Printf ("\n Enter search elements");
   Scanf ("%d ", & search);
   Pos = binary Search ( arr, size -1 , search);
   If ( Pos == -1) printf (" Not found \n") ;
```

```
else Printf ("\n the %d search elements is found at
                index %d \n", search, pos);
  return 0;
}
```

Output:

```
Enter the size of array (max 100) 5.
Enter the sorted elements in array
  5, 6, 7, 8, 9
Enter search element 8
the 8 search element is found at index 3
```