AP19110010477

ASSIGNMENT-6

SEARCHING AN SORTING in 'C'

CSE-G

SINDHU SANKATI

---

① Take the elements from the users 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 users 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+end)/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 ("\nEnter 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 element");
        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[ind1] + arr[ind2]);

printf("product is %d \n", arr[ind1]*arr[ind2]);

  return 0;

}
```

Output

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

Enter elements in array

5 2 3 6 7

The sorted array is:

7 6 5 3 2

Enter search elements

the 2 search element is found at index4

Enter two indexes

2 3

sum is 8

product is 15
```

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 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;
         merge sort (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 k th element is %d \n",
                product);
    return 0;
}
```

Output

```
Enter the size of the array 5
a[0]      = 1
a[1]      = 6
a[2]      = 1
a[3]      = 54
a[4]      = 2
Enter k
3

The product till the k th element is 2
```

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], insertion each element A[K] into its proper position in the previous sorted sub array A[1], A[2]... A[K-1].

Example :

array intial : 15, 19, 12, 21, 9

pass 1 : 15, 19, 12, 21, 9

pass2 : 12, 15, 19, 21, 9

pass3 : 12, 15, 19, 21, 9

pass 4 : 9, 13, 15, 19, 21 sorted.

sudo codo:

```
1. A[0] = minimum integer value
2. Repeat steps 3 through 8 for k=1,2,3,....., N-1
   {
3.   temp= A[K]
4.   ptr = k-1
3    Repeat steps 6 to 7 while temp < A[ptr]
     {
6      A[ptr+1] = A[ptr]
7      ptr = ptr - 1.
     }
```

```
    A[ptr+1] = temp
  }
END.
```

time complexity

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

space complexity: $O(1)$

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

Selection Sort.

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

example: 15, 6, 13, 3, 2 → smallest

pass 1. 2 15, 6, 13, 3, → smallest

pass 2 2, 3 15, 6, 13, → smallest

pass 3 2, 3, 6 15, 13 → smallest

pass 4 2, 3, 6, 13 15 → smallest

pass 5 2, 3, 6, 13, 15 → sorted.

pseudo code:

```
small = AR[1]
For i = 2 to n do {
    small = AR[i], pos = i
    For j = i to n do {
        If AR[j] < small then {
            small = AR[i], pos = (i)
        }
        j = j+1
    temp = AR(i), AR(i) = small, AR(pos) = temp)
}
END
```

Time complexity

best: $O(n)$

average $O(n^2)$

worst $O(n^2)$

space complexity

~~best~~ · $O(1)$

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, som=0, product=1;
    printf (" Alternate elements\n");
    for (i=0; i<size ; i++){
        if (i%2 != 0){
            product += arr[i];
        }
        else{
            som += arr[i];
            printf ("%d" arr [i]);
        }
    }
    printf ("\n Sum of the odd elements =%d \n", som);
    printf("\n Sum 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);
```

Scanned with CamScanner

```
        printf ("Elements divisible by %d \n", m);
        for (i=0; i< size; i++){
              if (arr[i] % m == 0)
                     printf ("%d ", arr[i]);
        }
}
void  bubbleSort (int arr[], int size)
{  int  i, j, temp;
     for (i=0; i<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(max100)");
   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

9
4
5
2
8

Alternate elements

2  5

Sum of the odd elements =7

product of the even elements =14

Enter the m

3

Element divisible by 3
9.
```

5) Write a recursive program to implement binary search?

```
# include <stdio.h>

int binarySearch (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 binarySearch (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 ("\n Enter sorted elements in array \n");
    for (i=0; i< size; i++){
        scanf ("%d, &arr[i]);
    }
printf ("\n Enter Search element");

scanf ("%d ", & search);

pos = binary Search (arr, 0, size -1, search);

if (pos == -1) printf ("Not found \n");

else printf ("\n the %d search element is found
 at index %d\n", search, pos);

 return 0;

}
```

<u>Output</u>

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

k .

Enter sorted elements in array

 1  2  3  4 5

enter search element 2

the 2 search element is found at index 1.
```