Assignment - 6

Navya Sadineni
AP19110010417
CSE-H.

1. Take the elements from the user and short them in decreasing order and do the following.

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

(b) The user to enter any two locations print the sum and product of values at those locations in the sorted array.

Sol:
```
# include <stdio.h>
int main( )
{
  int low, high, mid, n, key, arr[100], temp, p, one, two, sum,
      product
   printf (" enter the number of elements in array ");
   Scanf (" %d ", &n);
   printf (" enter %d integers,"n);
     for (i=0; i<n; i++)
     {
        if (i=0; i<n; i++)
        {
        if ( j= i+1 ; j<n; j++)
        {
          if (arr[i] < arr[j])
          {
           temp = arr[i];
```

```
            arr[i] = arr[j];
            arr[j] = temp;
          }
        }
     }
   printf("In element of array is sorted in descending
                order");
    for(i=0; i<n; i++)
     {
        printf("%d", arr[i]);
     }
     printf("Enter value to find");
     scanf("%d", & key);
     low = 0;
     high = n-1;
     mid = (low+high)/2;
     while(low< = high){
     if (arr[mid] > key)
        low = mid+1;
       else if (arr[mid] = = key){
       printf("%d found at location %d", key, mid+1);
       break;
       }
       else
       high = mid-1;
       mid = [low+high]/2;
```

```
}
if (low > high)
{
   printf ("Not found ! %d isn't present in the list,n, key);
}
   printf ("\n");
   printf (" enter two locations to find sum and product
                     of the elements ");

   scanf (" %d", &one);
   scanf (" %d", &two);
    sum = (arr[one] + arr[two]);
    product = (arr[one] arr[two]);
    printf ("The sum of elements = %d", sum);
    printf (" The product of elements = %d", product);
    return 0;
   }
```

Output :-

```
Enter number of elements in array 5
Enter 5 integers 9
7
5
4
2
Element of aray is sorted in decreasing order;
    9 7 5 4 2 enter value to find 5
    5 found at location 3
```

```
Enter two locations to find sum and product of the
elements 1
         3

The sum of elements = 14
The product of elements = 45.
```

Q. Write a c program for Merge Sort

```
# include< stdlib.h>
# include <stdio.h>
void merge( int arr[ ], int l, int m, int r)
{   int i,j,k;
    int n1 = m-l+1 ;
    int n2 = r-m;
    int L[n1], R[n2];
    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 first subarray
    j=0//; initial index of second 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++;
}
while (i<n1)
{
    arr[k]= L[i];
    i++;
    k++;
}
while (j<n2)
{
    arr[k]= R[j];
    j++;
    k++;
  }
}
void mergesort (int arr[], int l, int r)
{   if (l<r)
    {
```

```
    mergesort(arr, l, m);
    mergesort (arr, m+1, r);
     merge(arr, l, m, r);
   }
}
 void printArray(int A[i], int size)
{
      int i;
      for(i=0; i<size; i++)
            printf ( " %d", A[i]);
        printf("\n");
   }
   int main( )
   {
       int arr[5];
       int i;
       int arr_size = size of (arr)/ size of (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(" Insorted array is \n");
        printArray (arr, arr_size);
```

```
int k;
printf("enter the value of k");
Scanf("%d", &k);
int from_first = arr[k-1];
int from_last = arr[5-(k)];
printf("%d", fromlast * fromfirst);
return 0;
}.
```

3) Discuss insertion Sort and Selection Sort with examples.

Ans:- Definition of insertion sort :-

Insertion Sort 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 while 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.

Working of insertion Sort :-

→ It uses two Sets arrays where one Stores the Sorted data and other on unsorted data

* The sorting algorithm works until there are elements in the unsorted sets.

* Let's assume there are n number elements in the array initially the element with index 0((B=0) exists in the sorted set remaining elements are in the unsorted position of the list.

* the first element of the unsorted partion has array index 1 (If (B=0))

* After each iteration, it chooses the first element of c the inserted partition and inserts it into the proper place in the sorted set.

Advantages of insertion sort :-

-> Easily implemented and very efficient when used with small sets or data.

-> The additional memory space required of insertion sort is less (i.e; O(1)).

-> It is conducted to be live sorting technique as the list can be sorted as the new elements are received.

-> It is faster than other sorting algorithm.

complexity of insertion sort :-

the best case complexity of insertion sort is O(n).

worst case, running time of insertion sort is quadratic i.e. $O(n^2)$./n.

Example :-

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

Definition of Selection Sort :

The selection sort perform sorting by searching for the minimum value number and placing it into the first or last position according to the order. The process of searching minimum key and placing it in the proper position is continued untill the all elements are placed at right position.

Working of the Selection Sort :-

* Suppose an array ARR with N elements in the memory.
* In the first pass, the Smallest key is searched along with its position then the ARR[pos] is swapped with ARR[0]. Therefore, ARR[0] is Sorted
* In the second pass, again the position of the smallest value is determined in the subarray of N-1 elements interchange the ARR[pos] with ARR[1]
* In the pass N-1, the Same process is performed to sort the N number of elements.

Advantages of selection Sort :-

* The main advantages of selection Sort is that it performs well on a small list.
* Furthermore, because it is on in-place sorting algorithm, no additional temporary Storage is required beyond what is needed to hold the original list.

complexity of ~~inserti~~ Selection Sort :-

As the working of selection sort does not depend on the original order of the elements in the array, so there is no much difference between best case and worst case complexity of Selection Sort.

The running time complexity of Selection Sort

$= (n-1)+(n-2)+\dots+2+1 = n(n-1)/2 = O(n^2)$.

Scanned with CamScanner

example:-

0	1	2	3	4
17	16	3	13	6

Pass 1	17	16	3	13	6
	↑ min		↑ LOC		
pass 2	3	16	17	13	6
		↑ min			↑ LOC
Pass 3	3	6	17	13	16
			↑ min	↑ LOC	
Pass 4	3	6	13	17	16
				↑ min	↑ LOC
Pass 5	3	6	13	16	17

4)
```
# include <stdio.h>
void main( )
{ int a[100], n, i, j, temp, sum=0, prod= 1, m;
  printf(" enter number 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)
    {   sumo= sumo+a[i];
    }
}
printf("\nsum of odd index is %d", sumo);
for(i=0; i<n; i++)
{   if(i%2==0)
    {   prod = prod * a[i];
    }
}
printf("\nproduct of odd index is %d", prod);
printf("\nEnter 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 total number of elements to store : 5
Enter 5 elements : 8
6
4
3
2
Sorting array using bubble sort technique
All array elements sorted successfully!
Array elements in ascending order
2
3
4
6
8
Array elements in alternate order
2
4
8
The sum of odd position element are = 9
The product of even element are = 64.
```

5) # include <stdio.h>

```
int recursiveBinarysearch (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 recursiveBinarysearch (array, start_index,
                   middle-1, element);
     return recursive Binarysearch (array, middle+1,
     end_index, element) ;
  }
    return-1 ;
}
  int main(void)
 {
     int array[ ] ={ 1, 4, 7, 9, 16, 56, 70 };
     int n=7 ;
     int element =9;
     int found-index - recursive BinarySearch( 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:

```
Enter the size of array 5
Enter the values in sorted sequence
4
5
6
7
8
Enter a value to be search: 5
element is found at index 1
```