C-ASSIGNMENT-6

P.CHARAN
AP19110010038
CSE-G

1) Take the elements from the user and sort them in D.O and do the following

a) Using binearch search find the elements and the location in the array which the element is asked from user.

b) Ask the user to enter any two locations print the sum and product of values at these location in the sorted away.

```
#include <stdio.h>
int binary search (int arr[], int a, int b, int x)
{
if (b>=a){
int mid=a+ (b-a)/2;
if (arr[mid]==x)
return mid;
if (arr[mid]>x]
  return binary search (arr, a, mid-1, x);
return binarysearch (arr, mid+1, b, x);
}
return-1;
}
int main ()
{
  int num;
  printf ("Enter the size of array:");
  Scanf ("%d", &num);
  int i, j, a, val[num], op, var p1, p2, sum, pro;
```

```
for (a=0 ; a<num ; a++)
{
printf ("Enter Value");
scanf ("%d", &val[a]);
}
for (i=0; i<num; ++i)
{
for (J=i+1 ; J<num; ++J)
{
if (val[i] < val[J])
{
a=val[i];
val[i]=val[J];
val[J] =a;
}
}
}
printf("Array descending order:");
for(i=0; i<num ; i++)
{
printf("%d", val[i]);
}
printf("\n ** OPERATION_LIST **\n");
Printf ("1. find value at entered position \n
2. find the position of element \n 3. printing
sum & multiplication of values at entered
positions:");
printf ("\n Enter choice:\n");
Scanf ("%d", &op);
```

```
switch(op)
{
  Case: 1
  printf("Enter the position to obtain Value:");
  scanf("%d", &var);
  printf("The value at %d position is %d", var,
                        val[var]);
  break;
  Case: 2
  printf("Enter element of first position:");
  Scanf("%d", &var);
  int result = binary search(val, 0, num-1, var);
  (result == -1) ? printf("Elements is not
  present in array:");
  printf("Elements is present at index %d", result);
  return 0;
  Case: 3
  printf("\n Enter two positions to find sum and
          product of values \n");
  Scanf("%d %d", &p1, &p2);
  sum = val[p1] + val[p2];
  pro = val[p1] * val[p2];
  printf("sum = %d \n", sum);
  printf("MULTIPLICATION = %d", pro);
  break;
}
```

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

```
#include<stdio.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;
/*Create temp arrays */
int L[n1], R[n2];
/*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+l+j] ;
/* merge the temp arrays back into array*/
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++;
}

/* Copy the remaining elements of L[], if any
while (i < n1)
{
  arr[k] = L[i];
  i++;
  k++;
}

/* Copy the remaining elements of R[], if any
while(j < n2)
{
  arr[k] = R[j];
  j++;
  k++;
}
}
```

```
void
Voltage merge sort (int arr[], int l, int r)
{
 if (l<r)
 {
 int m = l+(r-l)/2;
 //Sort first and second halves.
 merged sort (arr, l, m);
 merge sort (arr, m+1, r);
 merge (arr, l, m, r);
 }
}
/* function to print an array */
 void print array (int A[], int size)
 {
 int i;
 for (i=0; i<size; i++)
  printf ("%d", A[i]);
  printf ("\n");
 }
 int main ()
 {
 int size, v;
 printf (" Enter array size :");
 scanf ("%d", &siz);
 int val [siz];
 for(v=0; v<=siz; v++);
 {
```

```
Printf ("Enter value :");
Scanf ("%d", &val[v]);
}
printf ("Given array is \n");
print array (val, size);
merge sort (val, 0, size-1);
printf ("\n sorted array is \n");
Print Array (val, siz);
int k, f, l, P1, P2 temp;
printf ("Enter the value of k to the find the
product of elements from first and last:");
Scanf ("%d", &k);
P1 = P2 = 1;
for (f=0; f<=k; f++)
{
temp = val[f];
P1 *= temp;
}
for (l = size-1; l>=k; l--)
{
temp = val[i];
P2 *= -temp;
}
printf ("product of kth element from first
and last are: %d %d", P1, P2);
}
```

3) Discuss insertion sort and selection sort with examples.

selection sort :-

selection sort selects ith smallest elements and places it at ith position. This algorithm divides the array into two parts. Sorted (left) and unsorted [Right] subarray. It selects the smallest element from unsorted subarray and places in the first position of that subarray (ascending order). It repetedly selects the next smallest element.

Time Complexity :-

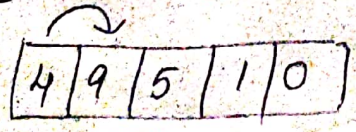
* Best Case [$O(N^2)$]. Also $O(N)$ swaps
* worst Case : Reversely sorted, and when inner loop makes maximum comparision : [$O(N^2)$]. Also $O(N)$ swaps.
* Average Case : [$O(N^2)$] : Also $O[N]$ swaps.

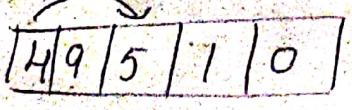
Space Complexity : (Auxiliary, $O[1]$) . In place sort

Steps and Iterations

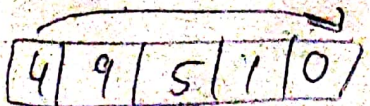
Selection Sort

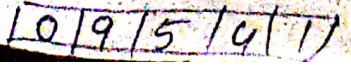
1st



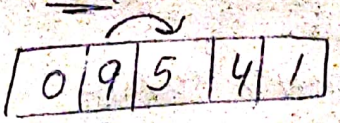



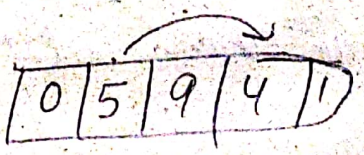


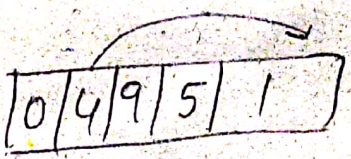


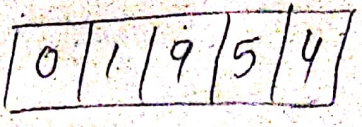




2nd.

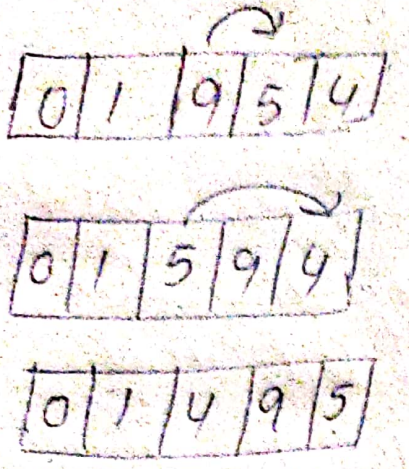


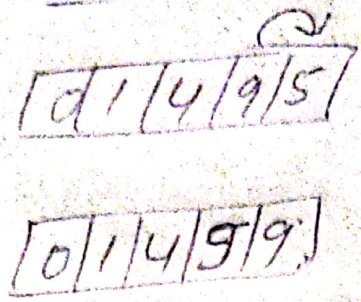



3rd




4th

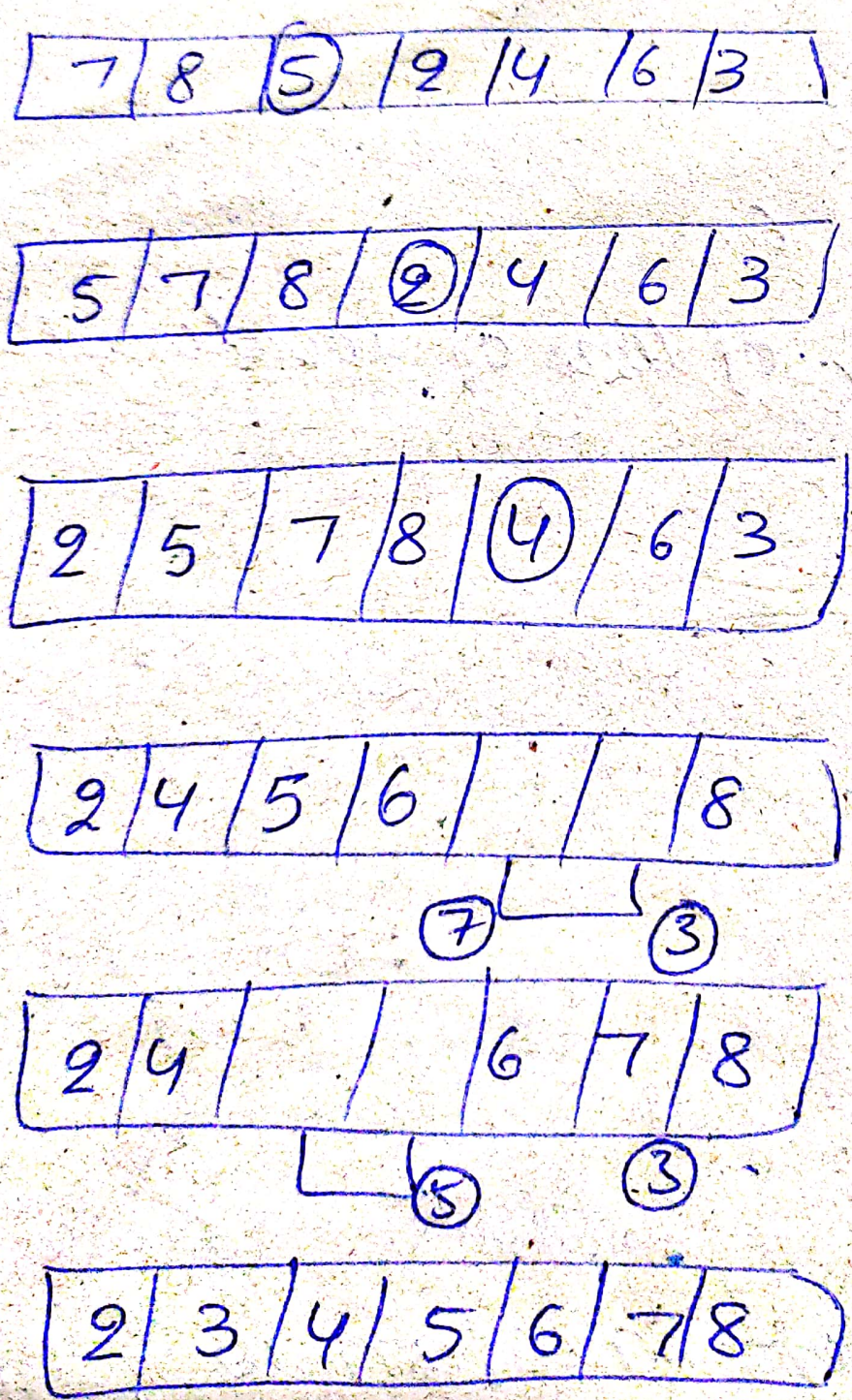



Insertion sort:- It is a simple comparision based sorting algorithm. It inserts every array element into its proper position. In ith iteration, previous (i-1) elements (i.e subarray Arr[1 : (i-1)] are already sorted and the ith element Arr[i]). Is inserted into its proper place in the previously sorted sub array.

Time Complexity:-

* Best Case:- Sorted array as input [O(N)] And O(1) swaps
* Worst Case; Reversely sorted and when inner loop makes maximum comparision [$O(N^2)$] And $O(N^2)$ swaps.
* Average case: [$O(N^2)$]. And $O(N^2)$ swaps.

Insertion Sort :- Ex:



4) Sort the array using bubble sort where elemen are taken from the user and display the elements
1) In alternate order
2) sum of elements in odd position and product of elements in even positions
3) elements which are divisible by M where M is taken from the user:

⇒
```
#include <stdio.h>
/* Bubble sort function */
void bubblesort (int ar[]; int n)
{
int i, J, temp;
for (i=0; i<n-1; i++)
for (J=0; J<n-i-1; J++)
if(ar[J]>ar[J+1]) /* Exchanging values
from using condition and temp variable*/
{
temp=ar[J];
ar[J]=ar[J+1];
ar[J+1]=temp;
}
}
int main()
{
int siz,i;
printf("Enter size of required array:");
Scanf ("%d",&siz);
int arr[siz];
```

```
for (i=0 ; i<siz ; i++)
{
  printf ("Enter element : ");
  scanf ("%d", &arr[i]);
}
bubble sort (arr, siz);
Printf ("Sorted array : \n");
for (i=0 ; i<siz ; i++)
{
  printf ("%d", arr[i]);
  printf ("\t");
}
printf ("\n **MENU**\n");
printf ("1. Display elements in alternate order\n");
printf ("2. sum of elements in odd positions and
        product of elements in even positions \n");
printf ("3. Divisible by m\n");
int op, sum =0, product =1, m;
printf (" Enter choice : ");
scanf ("%d", &op);
switch (op)
{
  case : 1 ;
  for (i=0; i<siz ; i+=2 )
  {
    printf ("%d\t", arr[i]);
  }
```

```
case 2:
for(i=0; i<siz; i+=2)
{
sum=sum+arr[i];
}
for(i=1; i<siz; i+=2)
{
product=product*arr[i];
}
printf("sum: %d\n",sum);
printf("product: %d\n",product);
case 3:
printf("Enter value m:");
scanf("%d",&m);
printf("Numbers divisible by %d are: \n",m);
for(i=0; i<siz; i++)
{
if(arr[i]%m==0)
{
printf("%d\t",arr[i]);
}
}
}
}
```

5) Write a recursive program to implement binary search ?

```
#include <stdio.h>
int binarysearch(int a[], int l, int h, int x)
{
int mid = (l+h)/2;
if (l>h) return -1;
if (a[mid] == x)
return mid;
if (a[mid] < x)
return binarysearch(a, mid+1, h, x);
else
return binary search(a, l, mid-1, x);
}
int main(void)
{
int a[100];
int siz, pos, val;
Printf("Enter lenght of array:");
Scanf("%d", &siz);
printf("\n Enter array elements \n");
for(int i=0; i<siz; i++)
Scanf("%d", &a[i]);
printf("Enter element of search \n");
Scanf("%d", &val);
pos = binary search(a,0,siz-1, val);
if (pos<0)
print("Cannot find element %d in array. \n", val);
else:
print("The position of %d in the array is %d. \n", val,
pos+1);
return 0
}
```