N. Lakshmi Manasa
AP19110010382
CSE-F

DS 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 print the sum and the product of values at those locations in the sorted array.

Program:-

```
#include<stdio.h>
int bisearch(int arr[], int top, int bot, int x)
{
if (bot >= top)
{
int mid = top + [bot - top]/2;
if (arr[mid] == x)
return mid;
if (arr[mid] > x)
return bisearch(arr, top, mid-1, x);
return bisearch(arr, mid+1, bot, x);
}
return -1;
}
int main()
```

Scanned with CamScanner

```
{
int num;
printf(" Enter the size of array: ");
scanf ("%d", &num);
int i, j, top, val[num], op, var, p1, p2, sum, pro;
for (top=0; top<num; top++)
{
printf(" Enter Value: ");
scanf ("%d", &val[top]);
}
for(i=0; i<num; i++)
{
for(j=0+i+1; j<num; j++)
{
  if (val[i]<val[j])
  {
    top=val[i];
    val[i]= val[j];
     val[j]= top;
  }
}
}
printf(" The array in descending order is : ");
```

Scanned with CamScanner

```
for(i=0; i<num; i++)
{
 printf("%d", val[i]);
}
printf("1. Find value at entered position\n 2. Find the position
        of element \n 3. Sum and multiplication of values at
        entered positions");
printf("Enter choice : \n");
scanf ("%d", &ch);
switch(ch)
{
case 1:
printf("Enter position to obtain value :");
scanf ("%d", &var);
printf("The value at %d position is %d", var, val(var));
break;
case 2:
printf ("Enter element to find position:");
scanf ("%d",&var);
int result = bisearch(val, 0, num-1, var);
(result==-1) ? printf("Element is not present in an array");
printf("Element is present at index %d", result);
return 0;
case 3:
printf("Enter two positions to find Sum and product of
                                        values\n");
```

Scanned with CamScanner

```
scanf ("%d %d", &p1, &p2);
sum = val[p1] + val [p2];
pro = val[p1] + val [p2];
printf (" sum is %d \n", sum);
printf (" Multiplication is %d ", pro);
break;
}
}
```

Output:-

```
Enter the size of array: 3
Enter value : 1
Enter value : 2
Enter value : 3
Array in descending order : 3 2 1
1. Find value at entered position
2. Find the position of element
3. Sum and multiplication of values at entered positions.
Enter choice : 3
Enter two positions to find sum and product of value
1
2
Sum is 3
Multiplication is 2.
```

Scanned with CamScanner

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.

Program:-

```
# include<stdio.h>
# include<stdlib.h>
void merge (int arr[], int a, int b, int c)
{
int i,j,k;
int n1= b-a+1;
int n2= c-b;
int L[n1], R[n2];
for(i=0; i<n1; i++)
L[i]= arr[a+i];
for(j=0; j<n2; j++)
R[j]= arr[b+1+j];
i=0;
j=0;
k=1;
while(i<n1 && j<n2)
{
if(L[n1]<=R[j])
{
```

Scanned with CamScanner

```
arr[k] = L[i];
i++;
}
else
{
arr[k]= R[j];
j++;
}
k++;
}
while (j<n2)
{
arr[k]= R[j];
j++;
k++;
}
}
void mergesort (int arr[], int a, int c)
{
if (a<c)
{
int b= a+(c-1)/2
mergesort (arr, a, b);
mergesort(arr, b+1, c);
merge (arr, a, b, c);
```

Scanned with CamScanner

```
    }
}

void printArray(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 size of an array: ");
  scanf("%d", &size);
  int val[size];
  for(v=0; v< size; v++)
  {
     printf("Enter Value: ");
     scanf("%d", &val[v]);
  }
  printf("Given array is \n");
  printArray(val, size);
  mergesort(val, 0, siz-1);
  printf("Sorted array is \n");
```

Scanned with CamScanner

```
printArray(val,size);
int K,f,a,P1,P2,temp;
printf("Enter the value of K to 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(i=size;a>=K;a--)
{
   temp=val[a];
   P2*=temp;
}
printf("Product of kth elements from first and last are: %d %d",P1,
                                                        P2);
}
```

Output:-

```
Enter array size : 5
Enter value : 1
Enter value : 2
Enter value : 4
Enter value : 35
```

Scanned with CamScanner

Enter value : 1

Given array is

1 2 4 35 1

Sorted array is

1 1 2 4 35

Enter the value of K to find the product of elements from first and last : 2

Product of kth elements from first and last are : 2 280

Scanned with CamScanner

3. Discuss Insertion sort and Selection sort with examples.

i. Insertion sort:-

A simple sorting algorithm that builds the final sorted array one at a time. It is in-place comparison-based algorithm.

A sub-list is maintained which is always sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place and has to be inserted there. Hence, it is called insertion sort.

Working:-

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list in the same array.

Ex:- Assuming an array

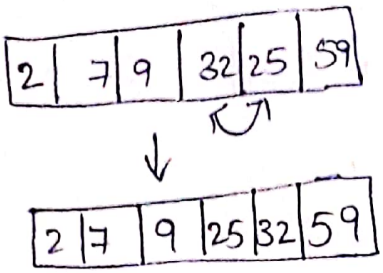
2	9	7	32	25	59

1. Comparing first two elements. 2,9 are in ascending order.

2	9	7	32	25	59

2. Moving ahead, comparing 9 and 7. As they are not in ascending order, we swap them.

Scanned with CamScanner



Swapping 32, 25

Insertion sort completed.

∴ Sorted array is 2 7 9 25 32 59

The algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$.

n = number of items.

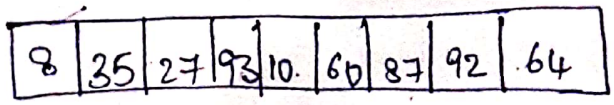
Best complexity: n

Space complexity: 1

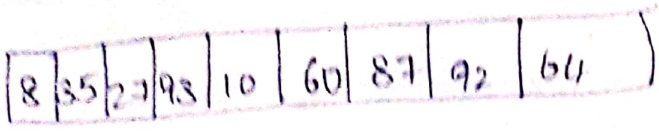
ii. Selection sort:-

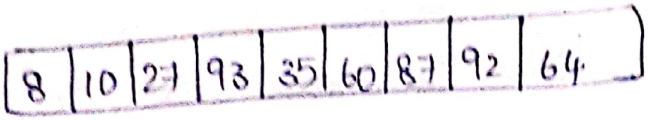
An in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list. The smallest element is selected from the unsorted array swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array bounded by one element to the right.

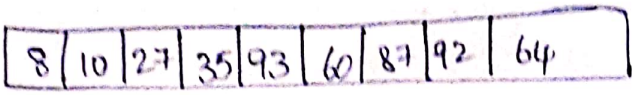
Working:-

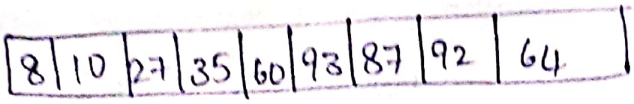


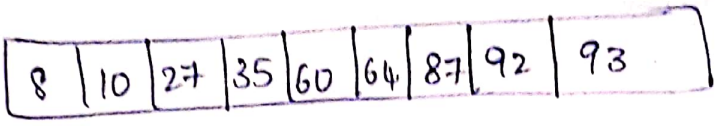
Scanned with CamScanner

2. 

3 

4 

5 

6 

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$

Best complexity :- $\Omega(n^2)$

Space complexity :- 1.

Scanned with CamScanner

4. Sort the array using bubble sort where elements are taken from the user and display the elements i. in alternate order
ii. Sum of elements in odd positions and product of elements in even positions
iii. Elements which are divisible by m where m is taken from the user.

Program:-

```
#include<stddio.h>
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++)
```

Scanned with CamScanner

```
if (arr[j] > arr[j+1])
{
temp = arr[j];
arr[j] = arr[j+1];
arr[j+1] = temp;
}
}
}
int main()
{
int size, i;
printf("Enter size of required array: ");
scanf("%d", &size);
int arr[size];
for(i=0; i<size; i++)
{
printf("Enter element: ");
scanf("%d", &arr[i]);
}
bubblesort(arr, size);
printf("Sorted array: \n");
for(i=0; i<size; i++)
{
 printf("%d", arr[i]);
 printf("\t");
```

Scanned with CamScanner

```
}
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");
int ch, sum=0; pro = 1; m;
printf ("Enter choice : ");
scanf ("%d", &ch);
{
 Case 1:
for(i=0; i< size; i+=2)
{
 printf ("%d\t", arr[i]);
 }
 Case 2:
for(i=0; i< size; i+=2)
{
sum = sum+arr[i]
}
for(i=1; i< size ; i+=2)
{
pro = pro * arr[i];
}
printf("Sum: %d\n", sum);
printf ("Product: %d\n", product);
```

Scanned with CamScanner

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

Output:-

```
Enter the size of required array: 5
Enter element : 1
Enter element : 2
Element Enter : 3
Enter Element : 5
Enter Element : 7
Sorted array:
1 2 3 5 7
/* MENU****/
1. Display elements in alternate order
```

Scanned with CamScanner

2. Sum of elements in odd positions and product of elements in even positions
3. Divisible by m

Enter choice : 2

Sum: 11

Product : 10

Enter choice : 3

Enter value of m: 3

Numbers divisible by 3 are : 3.

Scanned with CamScanner

5. Write a recursive program to implement binary search?

Program:-

```
#include<stdio.h>
#include<stdlib.h>
#define size 50
int bisearch(int[], int, int, int);
int main()
{
   int n, i, key, pos;
   int low, high, list[size];
   printf("Enter the total number of variables:");
   scanf("%d",&n);
   printf("Enter the number of elements in the list:\n");
   for(i=0;i<n;i++)
   {
      printf("Enter the value of a[%d]:",i);
      scanf("%d", &list[i]);
   }
   low=0;
   high=n-1;
   printf("Enter elements to be searched:");
   scanf("%d", &key);
   pos= bisearch(list, key, low, high);
   if (pos!=-1)
   {
```

Scanned with CamScanner

```
        printf("Number present at %d", (pos+1));
    }
    else
        printf("The number is not present in the list");
    return(0);
}

int bisearch(int a[], int x, int low, int high)
{
    int mid;
    if (low > high)
        return -1;
    mid = (low + high) / 2;
    if (x == a[mid])
    {
        return (mid);
    } else if (x < a[mid])
    {
        bisearch(a, x, low, mid-1);
    } else
    {
        bisearch(a, x, mid+1, high);
    }
}
```

Scanned with CamScanner

Output:-

```
Enter the total number of variables : 5
Enter the number of elements in the list : 5
Enter the value of a[0] : 1
Enter the value of a[1] : 2
Enter the value of a[2] : 3
Enter the value of a[3] : 4
Enter the value of a[4] : 5
Enter element to be searched : 4
Number present at 4.
```

Scanned with CamScanner