Y. Shiva prasad (1)
AP19110010242
CSE-G

Assignment-6

1.
```
#include<Stdio.h>
Void main( )
{
 int array[10], Sum, pro;
 int h,u,z,k, num,temp, keynum;
 int little, cen,rise;
 printf("Value of sort");
 Scanf("%d", &num);
 printf("Enter Elements");
 for(h=0;h<num;h++)
 {
 Scanf("%d", &array[h]);
 }
printf(" Enter Array Elements");
for(h=0;h<num;h++]{
printf("%d",array[h]);
}
for(h=0;h<num;h++)
{
for(u=0;u<(num-h-1);u++)
{
if(array[u]<array[u+1])
{
temp=array[u];
array[u]=array[u+1];
  array[u+1]=temp;}
}}
printf("Sorted array is");
for(h=0;h<num;h++){
```

Scanned with CamScanner

```
printf("%d", array[h]); }
printf(" Enter the elements to be searched");
scanf("%d", &keynum);
little=1;
rise=num;
do {
cen =(little+rise)/2;
if(keynum>array[cen])
rise=cen-1;
elseif (keynum>array[cen])
little=cen+1; }
while(keynum!=array[cen]&& little<=rise);
 if (keynum==array[cen])
 {
 printf(" element is founded", keynum, mid+1); }
else {
printf(" Search has failed"); }
printf("Enter the location");
scanf("%d %d", &z,&k);
z--;
k--;
for(h=0; h<num; h++) {
sum=array[z]+array[k];
 pro=array[z]* array[k]; }
printf(" Sum is", sum);
printf(" product is", pro); }
```

<u>Output:-</u>

```
Value of sort
 4
```

Scanned with CamScanner

```
Enterelements
 2
 3
 4
 5
Sortedarray is
5
4
3
2
Enter arrayelements
 4
element 3is found at 1
Enter the location
 1
 2
Sum is 3
Product is 2
```

2.

```
#include<stdio.h>
Void mergsort (intarray[ ],inti,intj);
Void merg(intarray[ ],inti1,intj1,inti2,intj2);
Void main( ){
int array[30],n,i,k;
printf("Enter the sort values");
Scanf("%d",&n);
printf("Enter values in array");
for(i=0;i<n;i++)
Scanf("%d",&array[i]);
mergsort(array,0,n-1);
printf("Sortedarray is");
for(i=0;i<n;i++)
```

Scanned with CamScanner

```
printf("%d",array[i]);
int bof=1,lof=1;
printf("Enter k value");
Scanf("%d",&k);
k=k-1;
for(i=0;i<=k;i++){
bof=bof*array[i];}
for(i=k;i<n;i++){
lof=lof*array[i];}
printf("product from first is",bof);
printf("product from last is",lof);
}
Void mergesort(int array[],int i,int j)
{
int mid;
if(i<j)
{
mid=(i+j)/2;
mergesort(array,i,mid);
mergesort(array,mid+1,j);
merge (array,i,mid,mid+1,j);}}
Void merge(int array[],int j1,int i1,int i2,int j2)
{
int temp[50];
int i,j,k;
i=i1;
j=i2;
k=0
while(i<=j1&&j<=j2)
{
```

```
if(array[i] < array[j])
temp[k++]=array[i++];
else
temp[k++]=array[j++];
}
while(i<=j1)
temp[k++]=array[i++];
while (j<=j2)
temp [k++]=array[j++];
for(i=i1, j=0; i<=j2; j++, i++)
array [i]= temp[j]; }
```

output:-

```
Enter the sort values 4
Enter values in array
15
26
12
11
Sorted array is 11 12 15 26
Enter the k value 12
The product from first is 144
The product from last is 51,480
```

3)

Insertion Sort:-

It is efficient for small data sets. It typically performs other simple quadratic algorithms, such as selection or bubble sort.

The time complexity is O(nk) when each element is at most k places away from its sorted position.

In works in a similar way as we arrange a deck of cards.

Eg:-

4 3 2 12 5

4 $\boxed{3}$ 2 12 5

3 4 $\boxed{2}$ 12 5

2 3 4 $\boxed{12}$ 5

2 3 4 12 $\boxed{5}$

2 3 4 5 12 → sorted array

Selection Sort:-

The selection sort algorithm sorts an array repeatedly finding the minimum element from unstorted part and putting it at the beginning.

Average & worst case complexity of this algorithm is O(n2)

Eg:-

3 9 27 16 1

Scan3, smallest1 Exchange

1 9 27 16 3

Exchange

1 3 27 16 9

Exchange

1 3 9 16 27

```
pseudo code
1. Small=np(L)
2. For I=2 to u do
3. Small=nR[i], pos=J
4. For 5= 1 To u do
5.    Small=AR[i], Pos[i]
6.    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)

```
#include<Stdio.h>
Void display Alt Sum(int arr[], int size){
int i, Sum=0, product=1;
printf("elements are alternate");
for(i=0; i<size; i++)}
      if(i%2 !=0)}
            product +=arr[i];}
   else{
        Sum+=arr[i];
        printf("%d", arr[i]);}}
printf(" Sum of odd elements =%d", Sum);
printf(" Sum of even elements =%d", product);
```

```
Void div(int arr[], int size){
     int i=0,m;
     printf("Eter them");
     Scanf("%d",&m);
     printf("elements divisible by%d",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; 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;}
    display Alt Sum pro(arr,size);
     divm(arr,size);}
  int main(){
    int arr[100], size,i;
    printf("size of array(max 100)");
     Scanf("%d",&size);
    printf("enter elements");
     for(i=0;i<size;i++){
         Scanf("%d",&arr[i];}
bubble sort(arr,size-1);
return 0; }
```

output:-

```
Enter the size of array(max 100)  5
Enter the elements in array
 10
 5
 6
 3
 9
Alternate elements are
  3  6
sum of odd elements = 17
Sum of even elements =16
enter the m
3
elements divisible by 3
9
```

5)
```
#include <stdio.h>
#include <stlib.h>
int Binary search(int arr[], int num, int start,
                  int last) {
 if(start>last)
  printf(" entered element is not found"); }
  else{
  int mid;
 mid=(start+last)/2;
if (arr [mid]==num) {
```

Scanned with CamScanner

```
printf("Elements desired is found at index %d",mid);
exit(0);}
else if(arr[mid]>num){
Binary Search (arr,num,first,mid-1);}
else{
Binary search(arr,num,mid+1, last); }}}
int main(){
int arr[]={10,82,65,139,145};
 int num=139;
 int start=0; last={sizeof(arr)/sizeof arr[0]})-1
  Binary search (arr,num,start,last);
 }
```

Output:-

```
Elements desired is found at index 3
```