1. Take the elements from the user and sort them in descending older 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. Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array.

→

```
# include <stdio.h>
void main ()
{
int a[30];
int i,j,a,n;
printf ("Enter size")
Scanf ("%d", &n);
printf ("Enter Elements ");
for (i=0;i<n;++i)
Scanf ("%d", &a[j]);
for (i=0;i<n;i++).
{
for (j= i+1;j<n;++j)
{
if (a[i] < a[j])
{
  a = a[i];
  a[i] = a[j];
  a[j] = a;
}
}
}
  printf(" descending order");
  for (i=0;i<n;i++).
  { printf("%d", a[i]);
  }
  int c, first, last, mid, s, l1, l2 Sum=0, P=1;
  printf("Enter Element");
  Scanf("%d, &s);
  first = 0;
  last = n-1;
```


Scanned with CamScanner

```
first =0;
 Last =n-1;
mid =(First+last)/2;
while (First<=last)
{
 if(a[mid]<search)
  first = middle+1;
else if (a[mid]==search)
   printf("%d found at %d", s, mid+1);
    break;
}
  else
    last = mid-1;
    mid = (first+last)/2;
 }
 if (first > last)
 { printf ("Not found");
 }
  printf("Enter two locations");
  scanf ("%d %d ", &l1, &l2);
  for(i=l1, i<l2 ; i++).
  {
     sum = sum+a[i];
      P = P+a[i];
  }
  printf("sum =%d", sum);
  printf("product =%d",p);
}
```

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.

→
```
# Include<stdio.h>
# Include<conio.h>
Int a[20], n, j;
 void sort(Int, Int), loco, high, mid, b[20];
 Void Merge (Int, Int, Int);
 void products;
  Void main()
```


Scanned with CamScanner

```
{
clrscrn();
printf("tutor size ");
scanf("%d",&n);
printf("Enter Elements");
for(i=0;i<n;i++).
   scanf("%d",&a[i]);
 low=0; high=n-1;
sort(low, high)
 printf("After sorting");
 for(i=0;i<n;i++)
 printf("%4d",a(i));
product();
getch();
}void sort(int low,int high)
{mid=(low+high)/2;
 if(low<high)
{sort(low, mid);
 sort(mid+1, high)
merge(low, mid, high);
}
void merge(int low, int mid, int high).
{ int l1, l2;
for(l1=0, l2=mid, i=0; l1<=mid, l1+l2<=high; i++)
{ if(a[l1]<a[l2])
   b[i]=a[l1++];
   else
     b[i]=a[l2++];
  }
while(l1<=mid)
   b[i++]=a(l1++);
 while(l2<=high)
    b[i++]=a(l2++);
for(i=0; i<n; i++)
   a[i]=b[i];
}
```


Scanned with CamScanner

```
void product( );
{ int p = 1;
  int k;
  printf("Enter k")
  scanf ("%d", &k);
for (i=0; i<=k ; i++).
{
  p = p * i;
}
printf("%d", p);
}
```

3. Insertion sort - The data is sorted by insertion the data into a existing sorted file, The process followed is elements are known before while location to place them is searched. Best case complexity is O(n).

   selection sort :- The data is sorted by selecting and placing the consecutive elements in sorted location. The best case complexity is $O(n^2)$.

4) →
```
# include <stdio.h>
int main( )
{ int a[100], n, c, d, swap;
printf("Enter size");
scanf ("%d", &n);
printf("Enter elements");
for (c=0; c<n; c++)
{scanf ("%d ", &a[c]);
}
for (c=0; c<n-1; c++)
{ for (d=0; d<n; -c-1; d++)
if (a[d] > a[d+1])
{
  swap = a[d];
  a[d] = a[d+1];
  a[d+1] = swap;
  }
 }
}
```


Scanned with CamScanner

```
printf("bubble sorted ");
for(c=0;c<n;c++)
{printf("%d",a[c]);
}
(i) printf("alternate elements");
for(c=0;i<=n;c+=2).
{printf("%d",a[c]);
}int sum=0;p=1;
(ii) for(c=1;c<=n;c+=2)
{
p=p*a[c];
}
for(c=0;c==n;c+=2)
{
s=s+a[c];
}
printf("sum & product = %d/%d", sum, p);
(iii) int m;
printf("Enter m");
scanf("%d",&m);
for(c=0";c<=n;c++)
{ if(a[c]%m==0)
{
printf("%d",a[c]);
}
else.
}print("Not found");
```

5)

```
#include<stdio.h>
int s['int a[], int f; int L, int e)
if(L>=f)
{int m=(f+L)/2;
if(a[m]==e)
{
return m;
}
if[a[m] >e)
{
```


Scanned with CamScanner

```
    return BS(a, f, m-1, e);
    }
  return BS(a, m+1, l, e);
  }
  return BS(a, m+1, l, e);
}
 return -1;
}
int main(void)
{
  int a[] = {1,4,3,2,9}
 int n = 6;
 int e = 9;
 int p=BS(a, 0, n-1, e);
  if (p == -1)
 {printf("Not found")
 }
 else.
 {
 printf("found at %d", p);
 }
}
```

——— ✱ ✱ ✱ ———

Scanned with CamScanner