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 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>
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 binary search(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)
```


Scanned with CamScanner

```
{
for (j=i+1;j<num; ++j)
 {
  if (val[i] < val[j])
   {
     a = val[i];
    val[i] = val[j];
     val[j] = a;
   }
 }
}
Printf ("Array in 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:-
```


Scanned with CamScanner

```
for (j=0; j<n2; j++)
  R[j] = arr[m+1+j];
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)
```


Scanned with CamScanner

```
Printf("Enter element to find position:");
scanf("%d",&var);
int result = binary search(val,0, num-1,var);
  (result==-1)
  Printf("element is not present in array");
  Printf("element 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("MULTIPLICATION = %d",pro);
 break;
 }
}
```

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 <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];
```


Scanned with CamScanner

```
{
 int m = l+(r-l)/2;
 merge sort (arr, l,m);
 merge sort (arr,m+1,r);
  merge (arr, l,m,r);
 }
}
void PrintArray(int A[], int size)
{
 int i;
 for(i=0; i<size; i++)
  Printf("%od", A[i]);
  Printf ("\n");
 }
int main( )
{
 int size, v;
 printf("enter array size :");
 scanf("%od", &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, siz);
 merge sort(val, 0, siz-1);
 Printf("\n sorted array is \n");
 PrintArray(val, siz);
 int k, f, i, P1, P2, temp;
```


Scanned with CamScanner

```
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=siz-1; i>k; i--)
 {
  temp = val[i];
   P2 *= temp;
 }
 printf("product of kth elements from first and last are
       : %d %d", P1, P2);
}
```

3. Discuss Insertion sort and selection sort with examples.

Insertion sort:-

Insertion sort works 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 till whole array is sorted in same order. The primary concept behind insertion sort is each item into its appreciate place in the final list. The insertion sort method saves an effective amount of memory. The advantage of Insertion sort is it works until there are elements in the unsorted set. Easily Implemented and very efficient when used with small sets of data. It is faster than other sorting techniques.

Scanned with CamScanner

The best case complexity of insertion sort is O(n) times. i.e when the array is previously sorted.

For example:-

If we have the array as {40,10,50,70,30} and we apply insertion sort to sort the array, then the resultant array after each iteration will be as

original array : {40,10,50,70,30}

Array after first iteration is : 10→40→50→70→30

Array after second iteration is: 10→40→50→70→30

Array after third iteration is: 10→40→50→70→30

Array after fourth iteration is: 10→30→40→50→70

* Selection sort:-

Selection sort is another algorithm that is used for sorting. This sorting algorithm, iterates through the array and finds the smallest number in the array and swaps it with it first element if it is smaller than the first element. Next, it goes on to the second element and so on untill all elements are sorted.

Example of selection sort:-

consider the array: [10, 5, 2, 1]

The first element is 10. The next part we must find the smallest number from the remaining array. The smallest number from 5 2 and 1 is 1. so, we replace 10 by 1.

The new array is [1, 5, 2, 10] Again this process is repeated.

The run time complexity of selection sort is $O(n^2)$. Advantage of selection-sort is no additional storage is required beyond what is needed to hold the original list.

Scanned with CamScanner

Q. 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.

code:-

```
#include<stdio.h>
Void bubblesort(int ar[],int n)
 {
  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]);
```


Scanned with CamScanner

```
Printf("\t");
}
Printf("\n**MENU**\n");
Printf("1.Display Elements in alternate order\n");
Printf("2. sum of elements in odd position and product
      of elements in even positions\n");
Printf("3. Divisible by m\n");
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);
```


Scanned with CamScanner

```
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 low,int high,int x){
 int mid =(low+high)/2;
 if (low>high) return -1;
 if(a[mid]==x) return mid;
  if (a[mid]<x)
   return binarysearch (a, mid+1,high,x);
  else
    return binarysearch (a,low, mid-1,x);
   }
int main(void) {
 int a[100]; len,pos,search_item;
  printf("%d", &len);
  printf("Enter the array elements \n");
  for (int i=0; i<len;i++)
   scanf("%d", a[i]);
  printf ("Enter the element to search \n");
  scanf ("%d", &search_item);
   pos = binarysearch (a, 0, len-1, search_item);
 if (pos <0)
 printf("cannot find the element %d in the array.\n",search_item);
 else
 printf(" Position of %d in array is %d \n,search_item,pos+1);
 return 0;
}
```


Scanned with CamScanner