G. krishna vamsi
AP19110010356
CSE-H.

DSA ASSIGNMENT

① Take the elements from the use and sort them in degending order and do the following.

a. using binary search find the element 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 array size:");
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 in Descending order")
 for(i=0; i<num; i++)
 {
  printf("%d", val[i]);
 }
 printf("/*MENU*/\n");
```

```
printf("1. find value at entered position \n")
printf("2. find position of entered element\n");
printf("3. print sum of product of values of entered location");
scanf("%d", &op);
switch (op)
{
 case 1;
 printf("enter position value (index) to obtain element:");
  scanf ("%d", &val);
  printf ("the value at possible %d is %d", var, val[var]);
  break;
  case2:
  print ("enter element to find position:");
   scanf("%d", &var);
   int result=binary search(val, 0, num-1, var);
   (result==-1)? printf("element not found");
   : printf ("element found at index %d", result);
   return 0;
  case 3;
  print f ("enter two index values;");
   scanf ("%d%d", &p1, &p2);
   sum = val[p1] + val[p2];
   pro = val[p1]+val[p2];
   printf ("sum = %d \n", sum);
   break;
   }
}
```

(3) Sort the array using Merge sort where elements are taken from the user and find the product of k^{th} element 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;
   /* create temp arrays */
  int L[n1], R[n2];
   /* copy data to temp arrays */
  for (i=0; i<n1; i++)
  L[i] = arr[l+i];
  for (j=0; j<n2; j++)
   R[j] = arr[m+1+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++;
}
while(j<n2)
{
arr[k]=R[j];
j++;
k++;
}
void merge sort(int arr[], int l, int r]
{
if(l<r)
{
int m=l+(r-l)/2;
// sort first & second halves
merge sort(arr, l, m);
merge sort(arr, m+1, r);
merge (arr(l, m, r);
}
}
void print array (int A[ ], Pnt size)
{int i;
for(i=0; i<size; i++)
printf("%d",A[i]);
printf("\n");
}
```

```
int main()
{
 int siz, v;
 printf("enter array size: ");
 scanf("%d", &siz);
 int val[siz];
 for (v=0; v<siz; v++)
 {
  printf("enter value; ");
  scanf("%d", &val[v]);
 }
  printArray(val, siz);
  mergesort(val, 0, siz-1);
  printf("insorted array is \n");
  printArray(val, siz);
  int k, f, l, p1, p2, temp;
    printf("enter k value:");
    scanf("%d", &k);
    p1=p2=1;
    for(f=0; f<k; f++)
    {
     temp=val[f];
      p1=temp*p1;
    }
     for(l=siz-1; l; l>=k; l--)
     {
      temp=val[f];
      p2=temp*p2;
      printf("%d %d", p1, p2);
    }
```

(3) Discuss insertion sort and selection sort with examples.

Insertion sort works by inserting the values in the existing sorted file. It constructs sorted array while inserting single element of a time. This process continues till array is sorted.

Selection sort perform sorting by searching for the minimum value number and placing it into the first last position according to the order (ascending/descending). The process of searching the minimum key and placing it in the proper position is continued until the all the elements are placed at right position.

Advantages

Insertion sort;

→ easily implemented any very efficient when used with small data sets.
→ Best case complexity : O(n)
→ Fastes than other sorting algorithms.
→ Live sorting technique.

Selection Sort;

→ Easy/simple implementation.
→ useful when data set is less
→ can be used when memory is less

Examples:-

Insertion sort

25 15 30 9 99 20
15 25 30 9 99 20
15 25 30 9 99 20

selection sort.

	0	1	2	3	4
1→	17	16	3	15	6
2→	3	16	17	15	6
3→	3	6	17	15	6

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 element in even positions.

(iii) elements which are divisible by m where m is taken from the user.

```
#include <stdio.h>
 /* Bubble sort */
void buddle sort (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]) /* Exchange Value; */
 {
  temp=ar[j];
  ar[j]=ar[j+1];
  ar[j+1]=temp;
 }
}
int main()
{
 int size, i;
 printf("Enter size of required array; ");
 scanf("%d", &size);
 int arr[size];
```

```
for (i=0; i<siz ; i++)
{
printf ("%d", &arr[i]);
printf ("\t")
}
printf ("\n\t* MENU *\n\n");
printf ("1.Display element in alternate order\n");
printf ("2.Sum of odd position elements and product
                                    of even position elemen");

printf ("\n3. Divisible by m\n");
int op, sum=0, product=1, m;
print ("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]);
}
}
}
```

① waite a recursive program to implement binary search?

```
#include <stdio.h>
int binary search(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 binary search(a, mid+1, h, x);
else
return binary search(a, l, mid-1, x);
}
int main(void)
{
int a[100], siz, pos, val, i;
printf("Enter array size:");
scanf("%d", &siz);
printf("\n Enter array elements:\n");
for(i=0; i<siz; i++)
Scanf("%d", &a[i]);
print f("\n Enter array elements:\n");
for(i=0; i<siz; i++)
Scanf("%d", & val);
```

```
pos = binarysearch(a,0,siz-1,val);
if (pos<0)
  printf("can't find element %d in array\n",val);
else
  printf("the position of %d in array is %d\n",
                                  val, pos+1);
  return 0;
}
```