M.R.M. Sitarama Reddy
AP19110010363
CSE-F

DSA ASSIGNMENT

(8.) Take the elements from the user and sort them in descending 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 binarySearch(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 binarySearch(arr, a, mid-1, x);
    return binarySearch(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 & product of values at entered locations");
printf("\n Enter choice:\n");
scanf("%d",&op);
switch(op)
{
  case 1:
  printf("Enter position value(index) to obtain element:");
  scanf("%d",&var);
  printf("The value at position %d is %d",var,val[var]);
  break;
  case 2:
  printf("Enter element to find position:");
  scanf("%d",&var);
  int result=binarysearch(val,0,num-1,var);
  (result==-1)? printf("Element not found");
  :printf("Element found at index %d",result);
  return 0;
```

```
case 3:
printf("Enter two index values:");
scanf("%d %d",&p1,&p2);
sum = val[p1] + val[p2];
pro = val[p1] * val[p2];
printf("SUM = %d \n",sum);
printf("MULTIPLICATION = %d",pro);
break;
}
}
```

---

(2) Sort the array using merge sort where elements are taken from the user and find the product of k^{th} elements from first and last where k is taken from the user

```
#include <stdlib.h>
#include <stdio.h>
void mergeSort
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 mergeSort(int arr[], int l, int r)
{
  if(l<r)
  {
    int m = l+(r-l)/2;
    // Sort first & second halves
    mergeSort(arr, l, m);
     mergeSort(arr, m+1, r);
      merge(arr(l, m, r);
    }
  }
void printArray(int A[], int 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("\nSorted 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>=k;l--)
{
    temp=val[l];
    p2=temp*p2;
}
printf("Product of kth elements from first & last are:");
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 at a time. This process continues till array is sorted.

Selection Sort perform sorting by searching for the ~~maximum~~ minimum value number and placing it into the first ~~and~~ 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 and very efficient when used with small data sets
  - → Best case complexity : O(n)
  - → Faster 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
9	15	25	30	99	20
9	15	25	30	99	20
9	15	20	25	30	99

- Selection sort

	0	1	2	3	4
1→	17 min	16	3 loc	15	6
2→	3	16 min	17	15	6 loc
3→	3	6	17 min	15 loc	16
4→	3	6	15 min	17 loc	16
5→	3	6	15	16	17

(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>
/* Bubblesort */
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++)
   if (ar[j] >ar[j+1]) /* Exchange values */
   {
    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 ("%d", &arr[i]);
    printf (" \t");
  }
  printf ("\n /* MENU */\n");
```

```
printf("1. Display elements in alternate order\n");
printf("2. Sum of odd position elements and product of even position elements");
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]);
       }
    }

}
}
```

5. Write a recursive program to implement binary search?

```
#include<stdio.h>
int binarysearch (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 binarysearch (a,mid+1, h,x);
    else
      return binarysearch (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]);
   printf ("Enter element to search:\n")
   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;
}
```