DSA - Assignment-6

J Vamsi Krishna
AP19110010366
CSE-F

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 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.

Sol
```
#include <stdio.h>
int binary search (int arr[], int a, intb, int x)
{  if (b>=a)
   {
     int mid = a+(b-a)/2;
      if (arr [mid] == x)
      return mid;
    return binarysearch (arr, a, mid-1, x);
    return bineary 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 , val, P1, P2, sum, pro;
    for(a=0;a<num;a++)
   {
    printf("Enter the 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 decreasing order: ");
     for(i=0; i<num; i++)
    { printf(" %d", Val [i]);
    }
printf (" /* Menu */n ");
 Printf(" 1. Find Value at entered position");
 printf("2. Print sum and product of values at entered locations")
 printf(" 3. find position of entered element \n");
 printf(" \n Enter your choice: ");
   scanf (" %d", &op)
  switch (op)
 {
   case1:
   printf (" Enter the position value(index) to obtain element:");
     scanf( ("%d", &var);
   printf(" The value at position %d is %d, var, val [var]);
      break;
     case2:
   printf(" Enter the two index values");
     scanf (" %d %d , &P1 &P2);
       sum = val [P1]+val [P2];
         pro= val [P1]* val[P2];
         printf (" Sum = %d \n ", sum);
         printf(" product = %d", pro);
     break;
Case 3:
printf(" Enter the element to find position:");
 Sconf ("%d", & var);
 int result = binary search (Val, 0, num-1, var);
 (result == -1) ? printf(" Element not found");
   : printf (" Element found at index %d", result);
   returno;
   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 user.

Sol
```
 #include <stdio.h>
 #include <stdlib.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<n; i++)
     L[i] = arr [l+i];
      for(j=0; j<n2 ; j++)
     R[j] = arr [m+1+j];

      i = 0;
      j=0;
      K = 1;
```

```
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;
   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 ("%d", A[i]);
     printf ("\n");
   }
  int  main()
  {
    int size, v;
    printf ("Enter array size: ");
    scanf ("%d", &size);
    int val[size];
    for (v=0;v<size ; v++)
    {
        printf(" Enter value :");
        scanf ("%d", &val [v]);
    }
  printArray(val, size);
   mergesort (val, 0, size-1);
   printf("\n Sorted array is \n");
     print Array(val, size);
     int k, f, l, p1, p2, temp;
   print ("Enter the k value:");
     scanf ( "%d", &k);

     p1 = p2 = 1;
   for ( f=0; f <=k; f++)
   {
      temp=val[f];
        p1 = temp * p1;
   }

   for (l = size-1; ~~l>~~ l>=k; l--)
    {  temp = val [f];
         p2 = temp * p2;
    }
   printf(" Product of kth elements from first and
          last are: ");
   printf ("%d %d", p1, p2);
}
```

3) Discuss insertion sort and selection sort using examples.

Sol 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 minimum value number and placing it into the first and last position according to the order (increasing / decreasing). The process of searching the minimum key and placing it in proper position is continued until the all elements are placed at right position.

Advantages:-

Insertion sort -

-> Best case complexity: o(n)

-> faster than other sorting algorithms

-> live sorting techunique

-> Easily implempted any very efficient when used with small data sets.

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	20	30	99
9	15	20	25	30	99

selection Sort:-

1 → 17	16	3	15	6
2 → 3	16	17	15	6
3 → 3	6	17	15	16
4 → 3	6	15	17	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 position and product of elements in even positions
(iii) Elements which are divisible by m where m is taken from user.

sol

```
#include<stdio.h>
void bubblesort (int arr[], int n)
{  int i,j,temp;
   for(i=0; i<n-1; i++)
   for (j=0; j<n-1; j++)
   if (arr[j] > arr[j+i])
    {
       temp=ar[j];
       ar[j]=ar[j+1];
    ar[j+1]=temp;
   }
}
```

```
int main()
{ int size, i;
  printf("Enter the size of required array:");
   scanf( "%d", &size);
    int arr [size];
    for (i=0; i<size; 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("3. Divisible by m \n");
   int op, sum=0; product =1, m;
    printf ("Enter choice:");
     scanf ("%d", &op);
    switch (op)
 {
    case1:
    for(i=0; i<size; i+=2)
    {
       printf("%d", arr[i]);
    }
    case2:
    for(i=0; i< size; i+=2)
    {
       sum=sum + arr [i]
    }
     for (i=1 ; i<size; i+=2)
      {
        product = product * arr[i];
      }
```

```
        printf (" sum: %d ", sum);
         printf(" Product : %d , Product );

        Case3:
        printf (" Enter  the  value  m: ");
          Scanf( " %d", &m);
          printf(" Numbers  divisible  by  %d are : \n ", m);
       for( i=0 ; i<size ; i++)
         {  if( arr[i] % m==0)
            {
              printf(" %d", arr[i]);
            }
         }
     }
 }
```

5) write 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 binarySearch (a, mid+1, n, x);
      else
      return binarysearch (a, l, mid-1, x);
   }
   int main(void)
   {  int a[100], size, pos, val, i;
        printf(" Enter the array size:" );
```

```
    scanf("%d",size);
  printf(" Enter the array elements: \n");
   for(i=0;i<size;i++)
   scanf("%d",&a[i]);
   printf(" Enter the elements to search \n");
    scanf("%d",&val);
    pos= binary search (a, 0, size-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;
}
```