C - Assignment

K.S.R.Subhash
CSE-F
AP19110010945

1)
```
#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 binary search [arr, a, mid-1, x);
    return binarysearch (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", & var [a]);
   }
   for(i=0; i<num; ++i)
   { for (j=i+1; j<num, ++j)
      { if (val [i]<val [j])
           val[i] =val[j];
           val [j] =a;
      }
   }
printf("Array in descending order :");
for (i=0; i<num ; i++){
    printf("%d", val [i]);
}
printf(" 1. Find value at entered position\n 2. Find the position of elements\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:
      printf("Enter element to find position:");
      scanf ("%d", &var);
      int result = binarysearch (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];
      printf ("Sum = %d\n", sum);
      printf (" Multiplication = %d", pro);
      break;
      }
}
```

2)
```
#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;
   int  L[n1] , R[n2];
   for (i=0 ; i<n1 ; j++)
        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<n; &&j<n2)
{ if (L[i]<=R[j])
   {  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)
   {  int m=l + (r-l)/2 ;

         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]);
     }
    printf (" Given array is \n");
    printArray (val , siz);
    mergesort (val,0, siz-1);
     printf ("\n sorted array is \n");
     printArray [val, siz];
    int k, f, l , p1 , p2, temp;
    printf (" Enter the value of k, to find the product of elements from
                first and last : ");
     scanf ("%d", &k);
for (f=0; f<=k; f++)
{ temp = val[f];
    p1* = temp;
}
for (l=3-1; l>=k; l--)
{
  temp = val[l];
    p2* = temp;
}
printf ("product of kth elements from first and last are: %d %d ",p1,p2);
}
```

3) Insertion sort:

Definition: Insertion sort works by inserting the set of values in the existing sorted file. It set of values in the existing sorted file a single element at a time. This process continuous until whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its appropriate place in the final list the insertion sort method save an effective amount of memory.

Advantages of Insertion sort;

⇒ Easily implemented and very efficient when used with small sets of data

⇒ the additional memory space requirement of insertion sort is less (i.e. O(1))

⇒ It is considered to be live sorting techniques as the list can be sorted as the new elements are receives;

⇒ It is faster than other sorting algorithms.

Example:-

```
4   3 2  10  12   1 5 6
4   3  2  10  12  1 5 6
3  4  2  10  12  1 5 6
2   3 4  10  12  1  5 6
2   3 4 10  12 1  5 6
 1  2  3  4   10  12 5 6
 1  2  3 4   5   10 12 6
 1  2  3 4  5  6  10 12
```

selection sort

Definition

The selection sort perform sorting by searching for the minvalue number and placing it into the first (or) last position according to the order 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 of selection sort

=> suppose an array ARR with N elements in the memory

=> Simple to understand the sorting of elements doesn't depend on the initial arrangement of the elements.

example:

```
step =0

i=0    20    12    10    15    2     minvalue at index 1
i=1    20    12    10    15    2     min value at index 2
i=2    20    12    10    15    2     min value at index 3
i=3    20    12    10    15    2     minvalue at index 4

       2   10    12    15   20
              swapping
```

4)
```
#include <stdio.h>
Void bubble sort (int arr[] , int n)
{ int i, j, temp;
  for (i=0 ; j <n-1 ; i++)
  for (j=0 ; j <n-i-1 ; j++)
     if (ar [j] > arr [j+1] {
          temp = ar [j];
     ar[j] = ar [j+1]
     arr[ j+1] = temp
     }
  }
```

Scanned by CamScanner

```
int main()
{ int size, i;
  printf (" Enter size of required array: ");
  scanf ("%d", & size);
  int arr [size];
  for (i=0; i<size ; i++){
        printf (" Enter Element :");
        scanf ("%d", & arr [i]);
  }
  bubble sort [arr, size];
  printf ("Sorted array : \n");
  for (i=0 ; i<size; i++){
        printf ("%d", arr [i]);
        printf ("\t");
  }
 printf ("1. Display element in alternate order\n 2. sum of elements in odd position and
        product of elements in even position\n 3. Divisible by m \n");
 int op, sum=0 product = 1, M;
 printf ("Enter choice: ");
 scanf ("%d", op);
 switch(op)
 { Case 1:
        for (i=0; i<size; i += 2)
        {  printf ("%d \t", arr [i]);
        }
        break;
   Case 2:
        for (i=0 ; i< size ; i+= 2)
          {  sum = sum + arr [i];
          }
        for (i=1 ; i<size ; i+=2)
         {  product = product * arr [i];
         }
        printf ("The sum and products are respectively %d and %d", sum, product );
Case 3:-
    printf(" Enter valuem: ");
    scanf ("%d", &m);
    printf (" Number divisible by %d are :\n", m);
    for (i=0; i<size; i++)
    {  if ( arr [i]%m  ==0){
           printf ("%d \t ", arr[i]);
       }
    }
  }
}

7)  #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(){
        int a[100];
        int size, pos, val;
        printf (" Enter length of the array");
        scanf ("%d ", &size);
        printf ("\n Enter array elements \n");
```

```
for (int i = 0 ; i<siz ; i++)
   scanf ("%d", &a[i]);
printf ("Enter element to search\n");
scanf ("%d", &val);
pos = binary search (a,0, siz-1, val);

 if (pos <0)
     printf (" Cann't find the element %d in the array \n", val);
 else  printf (" The position of %d in the array is %d \n", val, pos);

 return 0;
 }.
```