Assignment - 6

J. OmAryanReddy
AP19110010351
CSE - F

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 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)
    {
        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 position");
    print f("\n Enter choice:\n");
    scan f("%d", &op);
    switch(op)
{
    case 1:
    print f("Enter the position to obtain value:");
    scan f("%d", &var);
    print f("The value at %d position is %d", var, val
                                                [var]);
    break;
    case 2:
    printf("Enter element to find position:");
    scan f("%d", &val);
    int result = binary search(val, 0, num-1, val);
    (result == -1) ? print f("Element is not present in
                              array")
    print f("Element is present at index %d", result);
    return 0;
```

Case 2 :

```
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("SUM = %d\n", sum);
printf("MULTIPLICATION = %d", pro);
break;
}
}
```

2.

```
#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 L[] and R[] */
   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++;
}
 {
    arr[k] = L[i];
    i++;
    k++;
}
 {
    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 and second halves
       merge sort (arr, l, m);
       merge sort (arr, m + 1, r);
```

```
        merge(arr, l, m, r);
    }
}

/* Function to print an array */
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]);
    }
    printf("Given array is\n");
    printArray(val, size);
    mergesort(val, 0, size-1);
    printf("\nSorted array is\n");
    printArray(val, size);
    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 %d", &k);
    p1 = p2 = 1;
    for (i=0; i<=k; i++)
    {
      temp = val[i];
        p1 *= temp;
    }
    for (i=size-1; i>=k; i--)
    {
      temp = val[i];
      p2 *= temp;
    }
    printf("Product of kth elements from first
                and last are : %d %d", p1, p2);
    }
```

3. Insertion Sort.

Insertion sort works by inserting the set of values in the existing sorted file. It constructed the sorted array by intersecting a single element at a time. This process continous until whole thing 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 same an effective amount of memory.

* 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 considerd to be live sorting techniques as the list can be sorted as the new elements are recieved.

Selection sort.

Selection sort peeform sorting by seaeching for the min value numbees and placing it into the first cors last position according to the order. The process of seaeching the minimum key and placing it in the proper position is continued until the all the elements are placed at right position.

* suppose an array ARR with N elements in the memory.

* simple to understand the sorting of elements doesn't depend on the initial arrangment of the elements.

4.

```
#include<stdio.h>

void bubble 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[i] > ar[j+1]) /* Exchanging values using
   condition and temp variable */
  { 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<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("\n*** MENU ***\n");
     printf("Display elements in alternate order\n");
     printf("Sum of elements in odd positions
                and product of elements in even
                positions\n");
     printf("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<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:\n");
  scanf("%d", &m);
  printf("Numbers divisible by %d are:\n",
  for (i=0; i<siz; i++)
  {
    if (arr[i]%m==0)
    {
      printf("%d\t", arr[i]);
```

5.

```
#include <stdio.h>
int binarysearch (int a[], int l, int h, int n)
{
    int mid = (l+h)/2;
    if(l>h)return -1;
if (a[mid] == n)
return mid;
if (a[mid] < n)
 return binarysearch (a, mid + 1, h, n);
 else
 return binary search (a, l, mid -1, n);
}
 int main (void)
  { int a[100];
    int siz, pos, val;
printf("Enter length of the array\n");
  scanf ("%d", &siz);
print f ("\nEnter array elements\n");
  for (int i=0; i<siz; i++)
      scanf ("%d", &a[i]);
  print f ("Enter element to search\n");
    scanf ("%d", &val);
  pos = binarysearch (a, 0, siz -1, val);
printf ("Cannot find the element %d in the array.\n",
                                        val);
  else
print f ("the position of %d in the array is %d.\n",
    val, pos+1);      return 0; }
```