DSA Assignment

V. Manas
AP19110010506
CSE - G

1.

```
#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 of entered position at");
    printf (" 2. Find position of entered element \n");
    printf (" 3. Print sum and product of values at entered
                                                locations ");
    printf ("\nEnter 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 = binary Search (val D , 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.

```
#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; // Intial Index of second subarray
    k=l; // Intial index of mergedsubarray
    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 sort (arr, l, m, r);
    }
}
void print Array (int A[], int size)
{
    int i;
```

```
for (i = 0 ; i< size ; i++)
    printf ("%d", A[i]);
    printf ("\n");
}
int main ()
{
  int size;
  printf ("Enter array size :");
  scanf ("%d", &size);
  int val[siz];
  for (v=0; v< siz ; v++)
  {
     printf ("Enter value :");
     scanf ("%d", &val[v]);
  }
  print Array (val,siz);
  Merge sort (val, 0, siz-1);
  printf ("\n sorted array is \n");
  print array (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 and
                                    last are :");
printf ("%d %d", P1, P2);
}
```

3. 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.

Advantages of Insertion sort :

1. Easily implemented any very efficient when

used with small data sets.

* Best case complexity : O(n)
* Faster than other Sorting algorithms
* Live Sorting technique

Example of insertion Sort :

15	5	20	1	77	70
5	15	20	1	77	70
5	15	20	1	77	70
1	5	15	30	77	70
1	5	15	30	70	77

Selection Sort :

It perform Sorting by searching for minimum value number and placing it into the first or 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 :-

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

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

```
#include <stdio.h>
/*Bubble sort*/
void bubble sort (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+1]).      /* Exchange values*/
  {
     temp = arr[j] ;
     arr[j] = arr[j+1];
     arr[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 alternative order \n");
  printf (" 2. sum of odd position element and product of even position element
                                                              ;
  printf (" 3. \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<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.
```
#Include <stdio.h>
int binary search (int a[], int l , 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]);
    printf("Enter element to search: \n")
    scanf("%d", &val);
    pos = binary search (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;
```