ASSIGNMENT-6

N. Sai Keerthi
AP19110010543
CSE-H.

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 the size of array:");
 scanf ("%d", &num);
 int i, j, a var[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; ++i)
    {
      if(val[i] < val[j])
      {
        a = val[i];
        val[i] = val[i];
        val[i]=a;
      }
    }
}
  Printf("Array in descending Order:");
  for(i=0; i<num; i++)
  {
    printf("%d", val[i]);
  }
  printf("\n** OPERATIONAL_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 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 %od position is %od ", var, val[var]);
  break;
case2:
for(i=0; i<n2 ; i++)
   R[i] = arr[m+1+i];
 while (i<n1 && j<n2)
 {  if (L[i] <= R[j])
      {  arr[k] = L[i];
        i++;
 }
 else
 {
 arr[k] = R[j];
 j++;
}
 k++;
}
 while (i<n1)
 {  arr[k] = L[i];
    i++;
    k++;
 }
```

```
while(i<n1)
{
   arr[k] = R[j]
    j++;
    k++;
   }
}
void  merge sort (int arr[], int l, int r)
{
  if (l<r)
  Printf(" Enter element to find position:");
  scanf ("%d", & var);
  int result = binary search (val, 0, num-1, var);
      (result == -1)
     Printf (" Element is not present in array");
     Print f (" Element is present at index %d", result);
     return 0;
case 3:
Printf ("\n Enter two positions to find sum and product of
                values in:");
 Scanf ("%d %d", &P1, &P2);
   Sum = val[P1] + val[P2];
    PrO = val[P1] * val[P2];
    Print f(" Multiplication = %d", PrO);
    break;
  }
}
```

(1).

```
#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;
int L[n1], R[n2];
  for (i=0; i<n1; i++)
    L[i] = arr[l+i];
{
  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 size, v;
 printf("enter array size:");
 scanf("%d", &size);
 int val[size];
 for(v=0; v<size; v++)
{
 Printf(" Enter value-:");
 scanf("%od", &val[v]);
}
 Printf(" Given array is in");
 Printf Array(val, size);
 merge sort(val, 0, size-1);
 Printf("\n sorted array is \n");
 Print Array(val, size);
 int k, f, i, p1, p2, temp;
Printf("Enter the value of k to find the product of elements
      -from font and last:");
 scanf("%od", &k);
   p1=p2=1;
 for(f=0; f<=k; f++)
 {
    temp = val[f];
    p1 *= temp;
 }
for(i=size-1; i>k; i--)
```

```
{
  temp = val[i];
  p2* = temp;
  }
  printf("Product of kth elements from fork 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 constructs the sorted array by inserting a single element at a time. This process continues till whole array is sorted in same order. The primary concept behind insertion sort is each item into its appreciate place in the final list. The insertion sort method saves an effective amount of memory. The advantage of Insertion sort is it works until there are elements in the unsorted set. Easily Implemented and very efficient when used with small sets of data. It is faster than other sod it is works until there are elements in the unsorted set. Easily implemented and very efficient when used with sam sets of data. It is farted ther oths sorting techniques.

The best case complexity of Insertion sort in O(n) times i.e.

when the array is previously sorted.

For example:

If we have the array as {40,10,50,70,30} and we apply insertion sort to sort the array, then the resultant array after each iteration will be as

original array: {40,10,60,70,30}

Array after first iteration is : 10→40→60→70→30

Array after second iteration is : 10→40→60→70→30

Array after third iteration is : 10→40→50→70→30

Array after fourth iteration is : 10→30→40→40→70

* Selection Sort:

Selection sort is another algorithm that used for sorting that sorting algorithm, iterates through the array and finds the smallest number in the array and swaps it with it first element if it is smaller than the first element. Next, it goes on to the second element and so on until all elements are sorted:

Example of Selection sort:

Consider the array: [10,5,2,1]

The first element is 10. The next part we must find the smallest number from the remaining array. The smallest

numbers from 52 and 1 is 1. So we replace 10 by 1. The new array is [1,5,2,10] Again this process is repeated. The run time complexity of selection sort is $O(n^2)$. Advantage of selection-sort is no additional storage is required beyond what is needed to hold the input to the original list.

(4).

```
#include <stdio.h>
void bubble sort(int arr[], int n)
{
    temp = arr[j];
    arr[j] = arr[j+1];
    arr[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]);
  }
```

```
Printf("\t");
}
Printf("\n** MENU** *\n");
Printf(" 1. Display elements in alternate odd position
                and product of elements is even positions \n");
Printf (" 2. Display elements in alternate order \n");
Printf("3. Divisible by m\n");
Printf("Enter choice:");
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)
   {
     product= Produd* arr[i];
   }
   }
Printf("Sum: %d\n" Sum);
Printf("Product: %\n" product);
```

```
for (i=0; i<siz; i++)
{
   if (arr[i] % m == 0)
    {
      printf ("%d\n", arr[i]);
    }
  }
}
```

⑤

```
#include <stdio.h>
int binary search (int a[], int low, int high, int x){
int mid = (low + high)/2;
if (low > high) return -1;
if (a[mid] == x) return mid;
if (a[mid] < x)
 return binary search (a, mid+1, high, x);
 else
  return binary seoren (a, low, mid-1, x);
}
int main (void){
int a [100]; len, POS, Search_item;
Printf ("%d", &len);
Print f (" Enter the array element \n");
for (int i=0; i<len; i++)
Scanf ("%d", a[i]);
Print (" enter the element to search \n");
Scanf ("%d", & search_item);
```

```
if (pos<0)
printf("Cannot find the element %d in the array.\n", search
                                                    -item);
else
printf("Position of %d in array is %d\n", search_item, pos+1);
return 0;
}
```