Assignment-6

K. Lakshmi Pallya
AP19110010032
CSE-9

(1)

```
#include <stdio.h>
Void main()
{
 int a[30];
 Int i, j, a, n;
 Printf("enter size");
 scanf("%d", &n);
Printf("Enter elements");
for(i=0; i<n; i++)
   scanf("%d", &a[i]);
for(i=0; i<n; i++)
{
 for(j=i+1; j<n; j++)
 {
   if(a[i]<a[j])
    {
      a=a[i];
      a[i]=a[j];
       a[j]=a;
    }
 }
 }
  Printf("descending order");
  for(i=0; i<n; i++)
  {
    Printf("%d", a[i]);
  }
  int p, first, last, mid, s, m1, m2, sum=0, i=1;
  Printf("Enter element");
  scanf("%d", &s);
  first=0;
  last = n-1;
  mid = (first+last)/2;
  while(first <=last)
  {
```

```
if(a[mid]<search)
   first=middle+1;
else if (a[mid] == search)
    printf("%d found at %d", s, mid+1);
    break;
}
else
{last=mid-1;
  mid=(first+last)/2;
 }
if(first>last)
{
  printf("not found");
}
Printf("enter two locations");
scanf("%d%d", &m1, &m2);
for(i=m1; i<=m2; i++)
{
  sum=sum+a[i];
    P = P*a[i];
}
Printf("sum=%d", sum);
Printf("product=%d", P);
}
```

Output:-

enter the size 5

enter elements: 40 20 30 50 10

Descending order: 50 40 30 20 10

Enter two locations: 1 3

Sum = 90

Product = 6000

(2) /* c Program for merge sort */

```
#include <stdlib.h>
#include <stdio.h>
void merge(int arr[], int l, int m, int r)   //First subarray is arr[l...m]
                                             //second subarray is arr[m+1...r]
{
  int i, j, k;
  int n1 = m-l+1;
  int n2 = r-m;
  int L[n1], R[n2]; /*create temp arrays*/
  for(i=0; i<n1; i++) /*copy data to temp arrays L[] & R[]*/
     L[i] = arr[l+i];
  for(j=0; j<n2; j++)
     R[j] = arr[m+1+j];
  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)  /* copy the remaining elements of L[],
                  if there are any */
  {
    arr[k]=L[i];
    i++;
    k++;
  }
  while(j<n2)  /* copy the remaining elements of R[],
                  if there are any */
  { arr[k]=R[j];
    j++;
    k++;
  }
}
void mergesort(int arr[], int l, int r)  /* l is for left index and
                                            r is right index of the
                                            sub array of arr to be
                                            sorted */
{ if(l<r)
```

```
{
  int m= l+ (r-l)/2;
  mergesort(arr, l, m); //sort first and second halves
  mergesort(arr, m+1, r);
  merge (arr, l, m, r);
  }
}
void PrintArray(int A[], int size) /* Function to print an array*/
{ int i;
  for(i=0; i<size; i++)
     Printf("%d", A[i]);
   Printf("\n");
}
int main()
{
  int arr[5];
   int i;
   int arr_size = sizeof(arr)/sizeof(arr[0]);
   for(i=0; i<arr_size; i++){
     Printf("enter the elements");
     scanf("%d", &arr[i]);
    }
   Printf("Given array is \n");
   PrintArray(arr, arr_size);
   mergeSort(arr, 0, arr_size-1);
   Printf("\n sorted array is \n");
   PrintArray(arr, arr_size);
   int k;
   Printf("enter the value of k");
   scanf("%d", &k);
  int fromfirst = arr[k-1];
  int fromlast= arr[5-(k)];
  Printf("%d", fromlast * fromfirst);
   return 0;
}
```

Output:-

Enter the elements 1

Enter the elements 9

Enter the elements 3

enter the elements 5

Enter the elements 6

Given array is

1 9 3 5 6

Sorted array is

1 3 5 6 9

Enter value of K2

18

Insertion sort :-

Insertion sort is basically insertion of an element from a random set of numbers, to its correct position where it should actually be, by shifting the other elements if required.

Best case complexity is O(n)

Example:-

12 11 13 5 6 Pick element arr[i] and insert it into sorted sequence arr[0....i-1]

i=1. since 11 is smaller than 12, move 12 and insert 11 before 12

11, 12, 13, 5, 6

i = 2. 13 will remain at its position as all elements in A[0....i-1] are smaller than 13

11, 12, 13, 5, 6

i = 3. 5 will move to the begining and all other elements from 11 to 13 will move one position ahead of their current position.

5, 11, 12, 13, 6

i=4. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.

5, 6, 11, 12, 13

Selection sort :-

The smallest element is exchanged with the first element of the unsorted list of elements (the exchanged element takes the place where smallest element is initially placed). Then the second smallest element is exchanged with the second element ~~is exchanged~~ of the unsorted list of elements and so on until all the elements are sorted.

Example:-

Entered elements : 22 10 60 80 20

step 1:- 10 22 60 80 20 (10 and 22 exchanged position)

step 2:- 10 20 60 80 22 (20 and 22 exchanged position)

step 3:- 10 20 22 80 60 (22 and 60 exchanged position)

step 4:- 10 20 22 60 80 (60 and 80 exchanged position)

```
{
int a[100], n, c, d, swap;
printf("enter size");
scanf("%d", &n);
printf("enter elements:");
for(c=0; c<n; c++)
{
scanf("%d", &a[c]);
}
for(c=0; c<n-1; c++)
{
 for(d=0; d<n-c-1; d++)
 {
  if(a[d]<a[d+1])
  {
   swap=a[d];
    a[d]=a[d+1];
    a[d+1]=swap;
  }
 }
}
printf("bubble sorted");
for(c=0; c<n; c++)
{
 printf("%d", a[c]);
}
i) Printf("alternate elements");
 for(c=0; c<=n; c+=2)
 {
  Printf("%d", a[c]);
 }
ii) int sum=0; p=1;
 for(c=1; c<=n; c+=2)
  {
   p=p*a[c];
  }
 for(c=0; c<n; c+=2)
  {
   s=s+a[c];
  }
```

```
Printf("sum and Product = %d %d", sum, P);
```

iii)
```
Int m;
 Printf("Enter m");
 Scanf("%d", &m);
 for(c=0; c<n; c++)
 {
  if(a[c]%m==0)
   {
    Printf("%d", a[c]);
   }
  else
    Printf("not found");
 }
```

Output:-

```
Enter number of elements
5
Enter 5 integers
1 9 4 7 6
Sorted list in ascending order:
1 4 6 7 9
the alternate order is 1 6 9
Sum of odd index is 11
Product of odd index is 54
Enter the value of m
2
4 6
```

(5)
```
#include <stdio.h>
int binarysearch (int a[], int f, int l, int c)
{
  if (l >= f)
  {
    int m = (f+l)/2;
    if (a[m] == c)
    {
      return m;
    }
    if (a[m] > c)
    {
      return binarysearch(a, f, m-1, c);
    }
    return binarysearch(a, m+1, l, c);
  }
  return -1;
}
int main(void)
{
  int a[] = {1, 4, 5, 2, 9};
  int n = 5;
  int c = 9;
  int p = binarysearch(a, 0, n-1, c);
  if (p == -1)
  {
    printf("not found")
  }
  else
  {
    printf("found at %d", p);
  }
}
```

output

found at 3