

ASSIGNMENT-6SEARCHING AND SORTING

- 1) Take the elements from the user & sort them in descending order & do the following.
- Using Binary search find the element & the location in the array where the element is asked from user.
 - Ask the user to enter any two locations print the sum & product of values at those locations in the sorted array.

```
#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 binarySearch (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", &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 positions");
printf("\n Enter Choice: ");
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];
pro = val[p1] * val[p2];
printf printf("SUM = %d\n", sum);
printf("MULTIPLICATION = %d", pro);
break;
}
}

```

OUTPUT :-

Enter the size of array : 5

Enter Value : 87

Enter Value : 22

Enter Value : 95

Enter Value : 56

Enter Value : 32

Array in descending order : 95 87 56 32 22

****OPERATION-LIST****

1. Find value at entered position
2. Find the position of element
3. Printing sum & multiplication of values at entered positions.

Enter Choice:

1

Enter the position to obtain value : 3

The value at 3 position is 32

Program-2 :-

Sort the array using Merge sort where elements are taken from the user & find the product of kth element from first & last where k is taken from the user.

```
#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; i++)
```

```
        L[i] = arr[l + i];
```

```
    for (j = 0; j < n2; j++)
```

```
        R[j] = arr[m + 1 + j];
```

```
    i = 0;
```

```
    j = 0;
```

```
    k = l;
```

```
    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 (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;
        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("%d", &val[v]);
}
printf("Given array is \n");
printArray(val, size);
mergeSort(val, 0, size-1);
printf("In Sorted 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", &k);
p1 = p2 = 1;
for (f=0; f<=k; f++)
{
    temp = val[f];
    p1 *= temp;
}
for (l=size-1; l>=k; l--)
{
    temp = val[l];
    p2 *= temp;
}
printf("Product of Kth elements from first
        and last are: %d %d", p1, p2);
}

```

OUTPUT:-

Enter array size : 6

Enter Value : 1

Enter Value : 2

Enter Value : 3

Enter Value : 4

Enter Value : 5

Enter Value : 6

Given array is

1 2 3 4 5 6

Sorted array is

1 2 3 4 5 6

Enter the value of K to find the product of elements from first and last : 5

Product of Kth elements from first and last are : 5 10 15 20 25 30

Program-3:-

Discuss Insertion sort and Selection sort with examples.

Insertion sort:-

One element from the array is selected & is compared to the one side of the array and inserted to the proper position while shifting the rest of the elements accordingly.

Example:-

The lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place & then it has

to be inserted there. Hence the name Insertion sort.

Selection sort :-

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty & the unsorted part is the entire list.

Example :-

Consider the array:

$[8, 4, 3, 2]$

The first element is 8. The next part we must find the smallest number from the remaining array. The smallest number from 4, 3 and 2 is 2. So, we replace 8 by 2.

The new array is $[2, 4, 3, 8]$. Again, this process is repeated.

Finally, we get the sorted array as $[2, 3, 4, 8]$

Program-4 :-

Sort the array using bubble sort where elements are taken from the user & display the elements.

i. Σ in alternate order.

ii. Sum of elements in odd positions & product of elements in even positions.

iii, Elements which are divisible by m where m is taken from the user.

```
#include <stdio.h>
```

```
void bubbleSort(int arr[], int n)
```

```
{
```

```
    int i, j, temp;
```

```
    for (i = 0; i < n - 1; i++)
```

```
        for (j = 0; j < n - i - 1; j++)
```

```
            if (arr[j] > arr[j + 1])
```

```
            {
```

```
                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]);
```

```
    }
```

```
    bubbleSort(arr, size);
```

```
    printf("Sorted array : \n");
```

```
    for (i = 0; i < size; i++)
```

```
    {
```

```
        printf("%d", arr[i]);
```

```
    } printf("\n");
```

```
    }
```

```
    printf("\n** MENU **\n");
```

```
printf("1. Display elements in alternate order\n");  
printf("2. Sum of elements in odd positions  
and Product of elements in even  
positions\n");
```

```
printf("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]);
```

```
}
```

```
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("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<size; i++)
```

```
{
```

```
if(arr[i] % m == 0)
```

```

    }
    printf("%d\t", arr[i]);
}
}
}

```

OUTPUT:-

Enter size of required array: 5

Enter element : 14

Enter element : 36

Enter element : 85

Enter element : 47

Enter element : 96

Sorted array:

14 36 47 85 96

/** MENU **/

1. Display elements in alternate order.

2. Sum of elements in odd positions and Product of elements in even positions

3. Divisible by m

Enter Choice : 2

Sum : 157

Product : 3060

Enter value m: 2

Numbers divisible by 2 are :

14 36 96

Program-5:-

Write a recursive program to implement binary search?

```

#include <stdio.h>

```

```

#define MAX_LEN 10

```



```

int binary-search-recursive(int I[], int arr
                           arrayEnd, int arrayBegin, int a)
{
    int m, pos;
    if (arrayBegin <= arrayEnd)
    {
        m = (arrayBegin + arrayEnd) / 2;
        if (I[m] == a)
            return m;
        else if (a < I[m])
            return binary-search-recursive(I, array
                                           arrayBegin, m-1, a);
        else
            return binary-search-recursive(I, m+1,
                                           arrayEnd, a);
    }
    return -1;
}

void read_list(int I[], int n)
{
    int i;
    printf("In Enter the elements : \n");
    for (i=0; i<n; i++)
        scanf("%d", &I[i]);
}

void print_list(int I[], int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d\t", I[i]);
}

void main()
{
    int I[MAX_LEN], num, ele, f, I1, a;
    int pos;
    printf("In Binary Search using Recursion
           method");
}

```

```

printf("In Enter the number of elements:");
scanf("%d",&num);
read-list(I,num);
printf("In Elements present in the list
      are: \n\n");
print-list(I,num);
printf("\n\n Enter the element you want
      to search: \n\n");
scanf("%d",&ele);
{
    printf("In Recursive method: \n");
    pos = binary-search-recursive(I,0,num,ele);
    if (pos == -1)
    {
        printf("Element is not found");
    }
    else
    {
        printf("Element is found at %d
              position", pos);
    }
}
}

```

OUTPUT :-

Binary search using Recursion method
Enter the number of elements : 5

Enter the elements:

157

268

876

579

974

Elements present in the list are :

157 268 876 579 974

Enter the element you want to search:

268

Recursive method:

Element is found at 1 position.