

Practical No. 05 Write a C/C++ Code to implement (With Practical example Implementation)

1) Binary Search

```
// C program to implement recursive Binary Search
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        // If the element is present at the middle
        // itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, r, x);
    }

    // We reach here when element is not
    // present in array
    return -1;
}

int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1)
        ? printf("Element is not present in array")
        : printf("Element is present at index %d", result);
    return 0;
}
```

Output:

```
PS C:\Users\vaishnavi> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc binary.c -o binary } ; if ($?) { .\binary }
Element is present at index 3
PS C:\Users\vaishnavi\Desktop>
```

Merge Sort:

```
/* C program for Merge Sort */
#include <stdio.h>
#include <stdlib.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
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 arr[l..r]*/
    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++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
}
```

```

/* Copy the remaining elements of R[], if there
are any */
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
}

/* l is for left index and r is right index of the
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

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

/* UTILITY FUNCTIONS */
/* 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");
}

/* Driver code */
int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");

```

```

    printArray(arr, arr_size);
    return 0;
}

```

Output:

```

PS C:\Users\vaishnavi\Desktop> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc merge.c -o merge } ; if ($?) { .\merge }
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
PS C:\Users\vaishnavi\Desktop>

```

Strasses Multiplication

```

#include<stdio.h>
int main(){
    int a[2][2], b[2][2], c[2][2], i, j;
    int m1, m2, m3, m4, m5, m6, m7;

    printf("Enter the 4 elements of first matrix: ");
    for(i = 0; i < 2; i++)
        for(j = 0; j < 2; j++)
            scanf("%d", &a[i][j]);

    printf("Enter the 4 elements of second matrix: ");
    for(i = 0; i < 2; i++)
        for(j = 0; j < 2; j++)
            scanf("%d", &b[i][j]);

    printf("\nThe first matrix is\n");
    for(i = 0; i < 2; i++){
        printf("\n");
        for(j = 0; j < 2; j++)
            printf("%d\t", a[i][j]);
    }

    printf("\nThe second matrix is\n");
    for(i = 0; i < 2; i++){
        printf("\n");
        for(j = 0; j < 2; j++)
            printf("%d\t", b[i][j]);
    }

    m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
    m2= (a[1][0] + a[1][1]) * b[0][0];
    m3= a[0][0] * (b[0][1] - b[1][1]);

```

```

m4= a[1][1] * (b[1][0] - b[0][0]);
m5= (a[0][0] + a[0][1]) * b[1][1];
m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]);
m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]);

c[0][0] = m1 + m4- m5 + m7;
c[0][1] = m3 + m5;
c[1][0] = m2 + m4;
c[1][1] = m1 - m2 + m3 + m6;

printf("\nAfter multiplication using Strassen's algorithm \n");
for(i = 0; i < 2 ; i++){
    printf("\n");
    for(j = 0;j < 2; j++)
        printf("%d\t", c[i][j]);
}

return 0;
}

```

Output:

```

PS C:\Users\vaishnavi\Desktop> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc stassens.c -o stassens }
Enter the 4 elements of first matrix: 2 3 4 5
Enter the 4 elements of second matrix: 2 1 3 1

The first matrix is
2      3
4      5
The second matrix is
2      1
3      1
After multiplication using Strassen's algorithm
13     5
23     9
PS C:\Users\vaishnavi\Desktop> 

```

Quick sort:

```

#include<stdio.h>
void quicksort(int number[25],int first,int last){
    int i, j, pivot, temp;
    if(first<last){
        pivot=first;
        i=first;
        j=last;
        while(i<j){
            while(number[i]<=number[pivot]&& i<last)
                i++;

```

```

        while(number[j]>number[pivot])
            j--;
        if(i<j){
            temp=number[i];
            number[i]=number[j];
            number[j]=temp;
        }
    }
    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);
}
}

int main(){
    int i, count, number[25];
    printf("How many elements are u going to enter?: ");
    scanf("%d",&count);
    printf("Enter %d elements: ", count);
    for(i=0;i<count;i++)
        scanf("%d",&number[i]);
    quicksort(number,0,count-1);
    printf("Order of Sorted elements: ");
    for(i=0;i<count;i++)
        printf(" %d",number[i]);
    return 0;
}

```

Output:

```

PS C:\Users\vaishnavi\Desktop> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc quick.c -o quick }
How many elements are u going to enter?: 5
Enter 5 elements: 3 4 5 9 1
Order of Sorted elements: 1 3 4 5 9
PS C:\Users\vaishnavi\Desktop> █

```