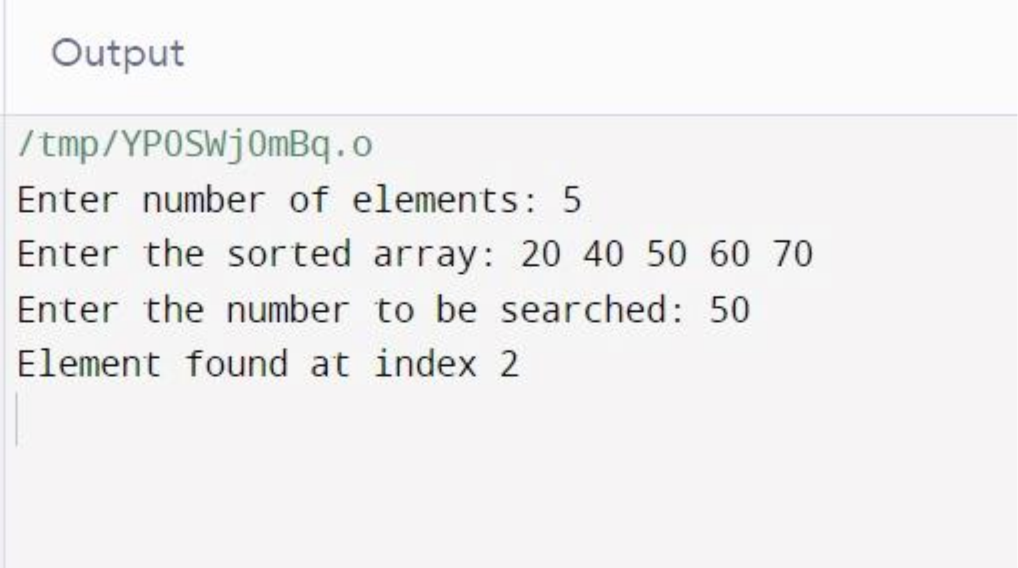
PRACTICAL NO- 05

Write a C/C++ Code to implement (With Practical example Implementation)

1) Binary Search

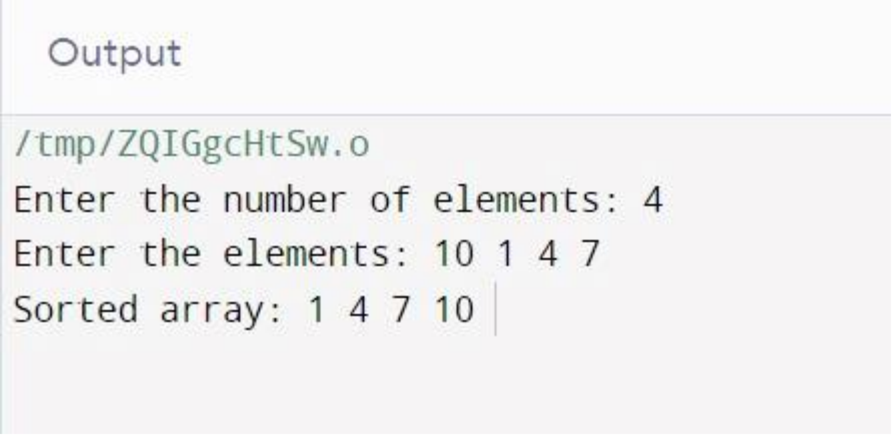
```
#include <iostream>

#include <vector>

int binarySearch(const std::vector<int> &arr, int x) {

    int left = 0, right = arr.size() - 1;

    while (left <= right) {

        int mid = left + (right - left) / 2;

        if (arr[mid] == x)

            return mid;

        if (arr[mid] < x)

            left = mid + 1;

        else

            right = mid - 1;

    }

    // Element was not found

    return -1;

}

int main() {

    int n, x;

    std::vector<int> arr;

    std::cout << "Enter number of elements: ";

    std::cin >> n;
    arr.resize(n);

    std::cout << "Enter the sorted array: ";

    for (int i = 0; i < n; i++)

        std::cin >> arr[i];

    std::cout << "Enter the number to be searched: ";

    std::cin >> x;

    int result = binarySearch(arr, x);

    if (result == -1)

        std::cout << "Element not found in the array." << std::endl;

    else

        std::cout << "Element found at index " << result << std::endl;

    return 0;

}
```



```
/tmp/YPOSWjOmBq.o
Enter number of elements: 5
Enter the sorted array: 20 40 50 60 70
Enter the number to be searched: 50
Element found at index 2
```

2) Merge Sort

```
#include<iostream>

using namespace std;

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;
    mergeSort(arr, l, m);
    mergeSort(arr, m+1, r);
    merge(arr, l, m, r);
  }
}
int main() {
  int n, i;
  cout<<"Enter the number of elements: ";
  cin>>n;
  int arr[n];
  cout<<"Enter the elements: ";
  for(i=0; i<n; i++) {
    cin>>arr[i];
  }
  mergeSort(arr, 0, n-1);
  cout<<"Sorted array: ";
  for (i=0; i < n; i++)
    cout<<arr[i]<<" ";
  return 0;
}
```

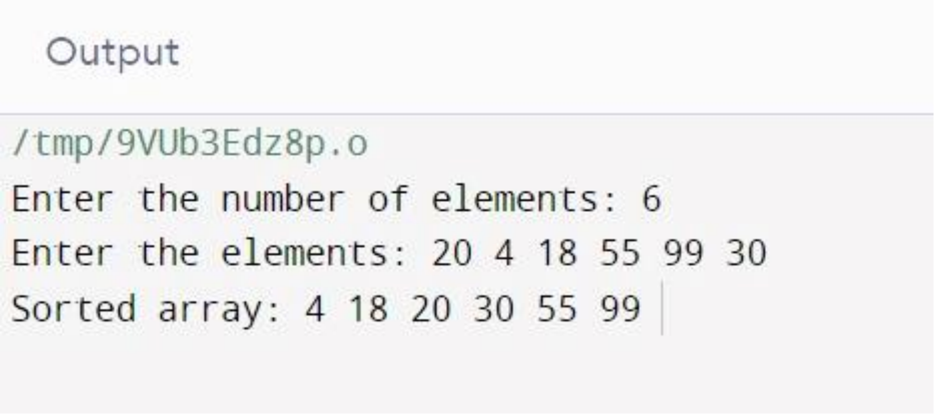


```
/tmp/ZQIGgcHtSw.o
Enter the number of elements: 4
Enter the elements: 10 1 4 7
Sorted array: 1 4 7 10
```

3) Quick Sort

```
#include<iostream>

using namespace std;

int partition(int arr[], int low, int high) {

  int pivot = arr[high];

  int i = (low - 1);

  for (int j = low; j <= high - 1; j++) {

    if (arr[j] < pivot) {

      i++;

      swap(arr[i], arr[j]);

    }

  }

  swap(arr[i + 1], arr[high]);

  return (i + 1);

}


void quickSort(int arr[], int low, int high) {

  if (low < high) {

    int pi = partition(arr, low, high);

    quickSort(arr, low, pi - 1);

    quickSort(arr, pi + 1, high);

  }

}
```

```
int main() {

  int n, i;

  cout<<"Enter the number of elements: ";

  cin>>n;

  int arr[n];

  cout<<"Enter the elements: ";

  for(i=0; i<n; i++) {

    cin>>arr[i];

  }

  quickSort(arr, 0, n-1);

  cout<<"Sorted array: ";

  for (i=0; i < n; i++)

    cout<<arr[i]<<" ";

  return 0;

}
```



```
/tmp/9VUb3Edz8p.o
Enter the number of elements: 6
Enter the elements: 20 4 18 55 99 30
Sorted array: 4 18 20 30 55 99
```

4) Strassen's Matrix multiplication

```
    #include <bits/stdc++.h>
    using namespace std;

    #define ROW_1 4
    #define COL_1 4

    #define ROW_2 4
    #define COL_2 4

void print(string display, vector<vector<int> > matrix,
        int start_row, int start_column, int end_row,
        int end_column)
{
    cout << endl << display << " =>" << endl;
    for (int i = start_row; i <= end_row; i++) {
        for (int j = start_column; j <= end_column; j++) {
            cout << setw(10);
            cout << matrix[i][j];
        }
        cout << endl;
    }
    cout << endl;
    return;
}

vector<vector<int> >
add_matrix(vector<vector<int> > matrix_A,
        vector<vector<int> > matrix_B, int split_index,
        int multiplier = 1)
{
    for (auto i = 0; i < split_index; i++)
        for (auto j = 0; j < split_index; j++)
```

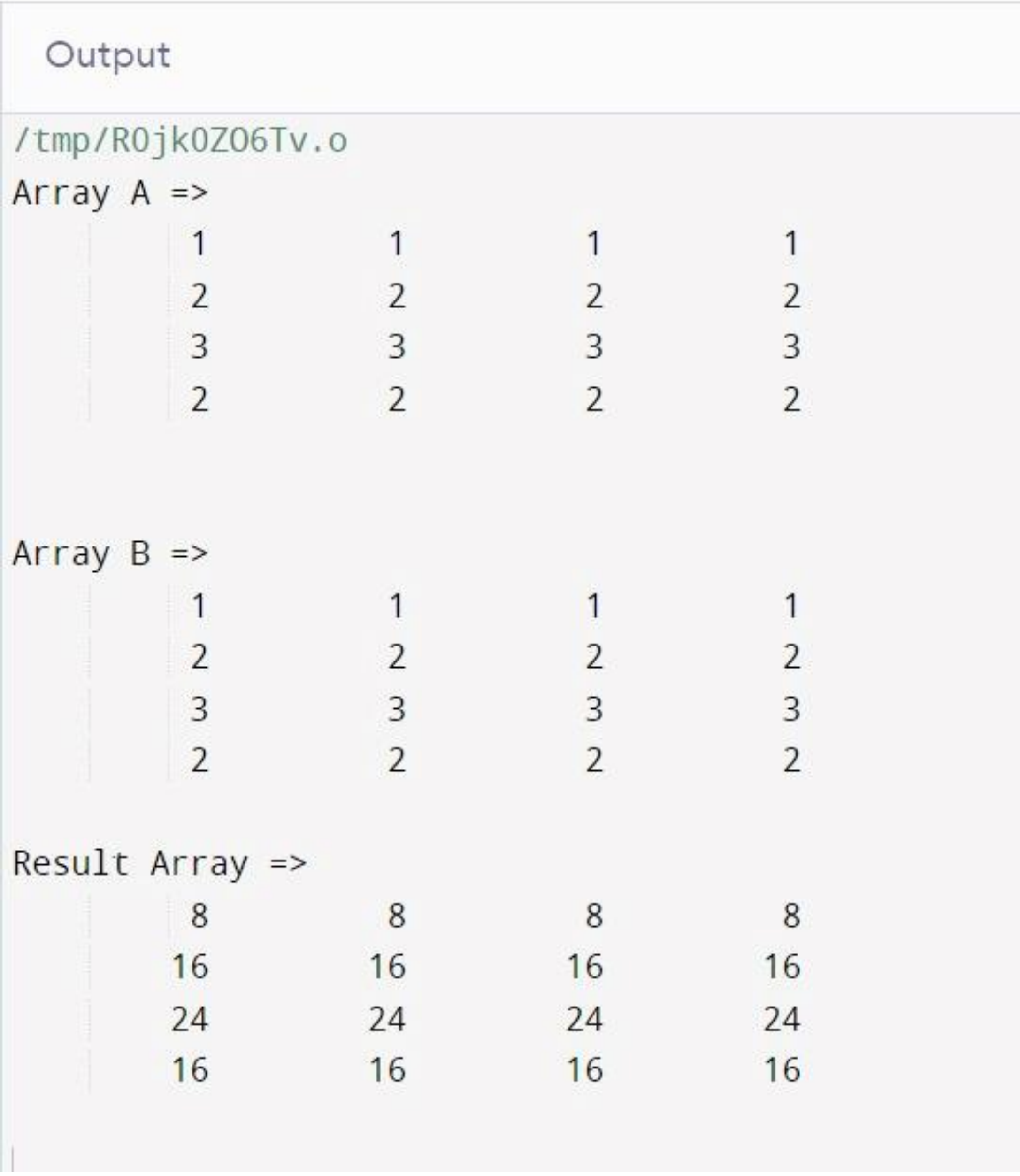
```
            matrix_A[i][j]
                = matrix_A[i][j]
                + (multiplier * matrix_B[i][j]);
    return matrix_A;
}

vector<vector<int> >
multiply_matrix(vector<vector<int> > matrix_A,
                vector<vector<int> > matrix_B)
{
    int col_1 = matrix_A[0].size();
    int row_1 = matrix_A.size();
    int col_2 = matrix_B[0].size();
    int row_2 = matrix_B.size();

    if (col_1 != row_2) {
        cout << "\nError: The number of columns in Matrix "
                "A must be equal to the number of rows in "
                "Matrix B\n";
        return {};
    }

    vector<int> result_matrix_row(col_2, 0);
    vector<vector<int> > result_matrix(row_1,result_matrix_row);

    if (col_1 == 1)
        result_matrix[0][0]
            = matrix_A[0][0] * matrix_B[0][0];
    else {
        int split_index = col_1 / 2;

        vector<int> row_vector(split_index, 0);

        vector<vector<int> > a00(split_index, row_vector);
```

```
vector<vector<int> > a01(split_index, row_vector);
vector<vector<int> > a10(split_index, row_vector);
vector<vector<int> > a11(split_index, row_vector);
vector<vector<int> > b00(split_index, row_vector);
vector<vector<int> > b01(split_index, row_vector);
vector<vector<int> > b10(split_index, row_vector);
vector<vector<int> > b11(split_index, row_vector);

for (auto i = 0; i < split_index; i++)
        for (auto j = 0; j < split_index; j++) {
                a00[i][j] = matrix_A[i][j];
                a01[i][j] = matrix_A[i][j + split_index];
                a10[i][j] = matrix_A[split_index + i][j];
                a11[i][j] = matrix_A[i + split_index][j + split_index];
                b00[i][j] = matrix_B[i][j];
                b01[i][j] = matrix_B[i][j + split_index];
                b10[i][j] = matrix_B[split_index + i][j];
                b11[i][j] = matrix_B[i + split_index][j + split_index];
        }

vector<vector<int> > p(multiply_matrix(
        a00, add_matrix(b01, b11, split_index, -1)));
vector<vector<int> > q(multiply_matrix(
        add_matrix(a00, a01, split_index), b11));
vector<vector<int> > r(multiply_matrix(
        add_matrix(a10, a11, split_index), b00));
vector<vector<int> > s(multiply_matrix(
        a11, add_matrix(b10, b00, split_index, -1)));
vector<vector<int> > t(multiply_matrix(
        add_matrix(a00, a11, split_index),
        add_matrix(b00, b11, split_index)));
vector<vector<int> > u(multiply_matrix(
        add_matrix(a01, a11, split_index, -1),
        add_matrix(b10, b11, split_index)));
```

```
vector<vector<int> > v(multiply_matrix(
    add_matrix(a00, a10, split_index, -1),
    add_matrix(b00, b01, split_index)));

vector<vector<int> > result_matrix_00(add_matrix(
    add_matrix(add_matrix(t, s, split_index), u,
    split_index),
    q, split_index, -1));
vector<vector<int> > result_matrix_01(
    add_matrix(p, q, split_index));
vector<vector<int> > result_matrix_10(
    add_matrix(r, s, split_index));
vector<vector<int> > result_matrix_11(add_matrix(
    add_matrix(add_matrix(t, p, split_index), r,
    split_index, -1),
    v, split_index, -1));

for (auto i = 0; i < split_index; i++)
    for (auto j = 0; j < split_index; j++) {
        result_matrix[i][j]
            = result_matrix_00[i][j];
        result_matrix[i][j + split_index]
            = result_matrix_01[i][j];
        result_matrix[split_index + i][j]
            = result_matrix_10[i][j];
        result_matrix[i + split_index][j + split_index]
            = result_matrix_11[i][j];
    }

a00.clear();
a01.clear();
a10.clear();
a11.clear();
b00.clear();
```

```
        b01.clear();
        b10.clear();
        b11.clear();
        p.clear();
        q.clear();
        r.clear();
        s.clear();
        t.clear();
        u.clear();
        v.clear();
        result_matrix_00.clear();
        result_matrix_01.clear();
        result_matrix_10.clear();
        result_matrix_11.clear();
    }
    return result_matrix;
}

int main()
{
    vector<vector<int> > matrix_A = { { 1, 1, 1, 1 },
                                      { 2, 2, 2, 2 },
                                      { 3, 3, 3, 3 },
                                     { 2, 2, 2, 2 } };

    print("Array A", matrix_A, 0, 0, ROW_1 - 1, COL_1 - 1);

    vector<vector<int> > matrix_B = { { 1, 1, 1, 1 },
                                      { 2, 2, 2, 2 },
                                      { 3, 3, 3, 3 },
                                     { 2, 2, 2, 2 } };

    print("Array B", matrix_B, 0, 0, ROW_2 - 1, COL_2 - 1);
```

```
        vector<vector<int> > result_matrix(
                multiply_matrix(matrix_A, matrix_B));

        print("Result Array", result_matrix, 0, 0, ROW_1 - 1,
                COL_2 - 1);
}
```



```
/tmp/R0jk0Z06Tv.o
Array A =>
                1       1       1       1
                2       2       2       2
                3       3       3       3
                2       2       2       2


Array B =>
                1       1       1       1
                2       2       2       2
                3       3       3       3
                2       2       2       2

Result Array =>
                8       8       8       8
               16      16      16      16
               24      24      24      24
               16      16      16      16
```