**EXPERIMENT-1**

Q) Write a program to implement binary search with recursion and without it

**PSEUDOCODE:**

function binarySearch(array, target):

start = 0

end = length of array - 1

while start is less than or equal to end:

middle = (start + end) divided by 2

if array[middle] == target:

return middle

else if array[middle] < target:

start = middle + 1

else:

end = middle - 1

return -1

**CODE;**

// WITHOUT RECURSION

#include <iostream>

using namespace std;

int binarySearch(int array[], int size, int target) {

int start = 0;

int end = size - 1;

while (start <= end) {

int middle = (start + end) / 2;

if (array[middle] == target)

return middle;

else if (array[middle] < target)

start = middle + 1;

else

end = middle - 1;

}

return -1;

}

int main() {

int size, target;

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

cin >> size;

int array[size];

cout << "Enter " << size << " sorted elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

cout << "Enter the number to search for: ";

cin >> target;

int result = binarySearch(array, size, target);

if (result != -1)

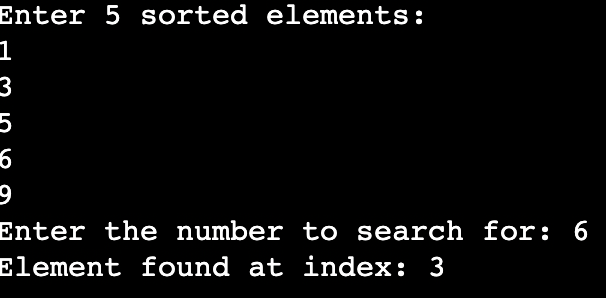
cout << "Element found at index: " << result << endl;

else

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

return 0;

}



Name:- Udit Bansal

A023166923097

// **WITH RECURSION**

#include <iostream>

using namespace std;

int binarySearch(int array[], int start, int end, int target) {

if (start > end)

return -1;

int middle = (start + end) / 2;

if (array[middle] == target)

return middle;

else if (array[middle] < target)

return binarySearch(array, middle + 1, end, target);

else

return binarySearch(array, start, middle - 1, target);

}

int main() {

int size, target;

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

cin >> size;

int array[size];

cout << "Enter " << size << " sorted elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

cout << "Enter the number to search for: ";

cin >> target;

int result = binarySearch(array, 0, size - 1, target);

if (result != -1)

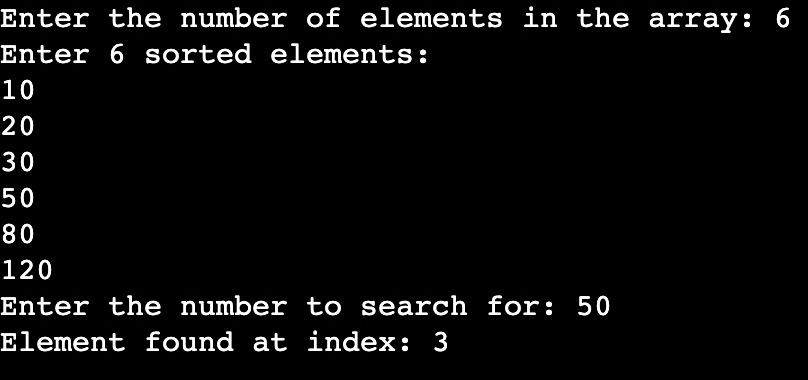
cout << "Element found at index: " << result << endl;

else

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

return 0;

}



Name:- Udit Bansal

A023166923097

**EXPERIMENT-2**

Q) Write a program to implement selection sort and bubble sort

**PSEUDOCODE:**

// SELECTION SORT

function selectionSort(array):

n = length of array

for i from 0 to n - 1:

minIndex = i

for j from i + 1 to n - 1:

if array[j] < array[minIndex]:

minIndex = j

if minIndex is not i:

swap array[i] and array[minIndex]

// BUBBLE SORT

function bubbleSort(array):

n = length of array

for i from 0 to n - 1:

for j from 0 to n - i - 2:

if array[j] > array[j + 1]:

swap array[j] and array[j + 1]

**CODE:**

// SELECTION SORT

#include <iostream>

using namespace std;

void selectionSort(int array[], int size) {

for (int i = 0; i < size - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < size; j++) {

if (array[j] < array[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

int temp = array[i];

array[i] = array[minIndex];

array[minIndex] = temp;

}

}

}

int main() {

int size;

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

cin >> size;

int array[size];

cout << "Enter " << size << " elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

selectionSort(array, size);

cout << "Sorted array:\n";

for (int i = 0; i < size; i++) {

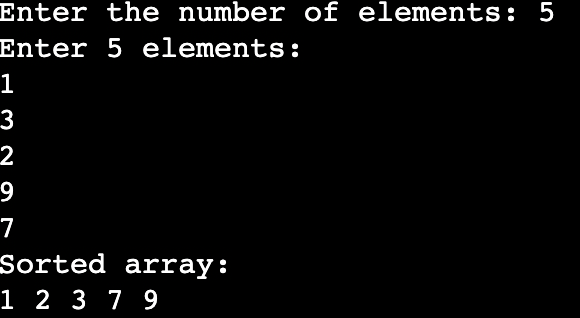
cout << array[i] << " ";

}

cout << endl;

return 0;

}



Name:- Udit Bansal

A023166923097

// BUBBLE SORT

#include <iostream>

using namespace std;

void bubbleSort(int array[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (array[j] > array[j + 1]) {

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

int main() {

int size;

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

cin >> size;

int array[size];

cout << "Enter " << size << " elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

bubbleSort(array, size);

cout << "Sorted array:\n";

for (int i = 0; i < size; i++) {

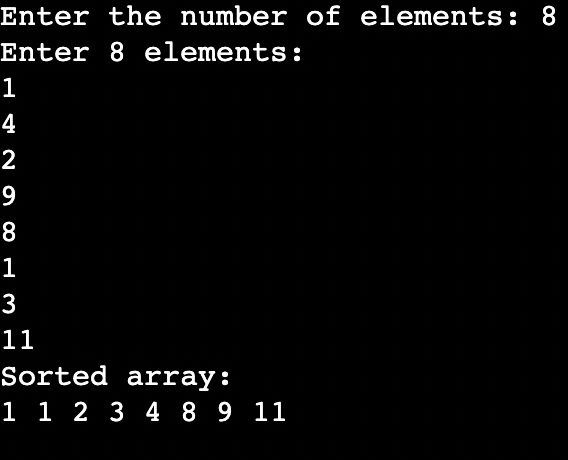
cout << array[i] << " ";

}

cout << endl;

return 0;

}



Name:- Udit Bansal

A023166923097

**EXPERIMENT-3**

Q) Write a program to implement quick sort

**PSEUDOCODE:**

function quickSort(array, low, high):

if low < high:

pivotIndex = partition(array, low, high)

quickSort(array, low, pivotIndex - 1)

quickSort(array, pivotIndex + 1, high)

function partition(array, low, high):

pivot = array[high]

i = low - 1

for j from low to high - 1:

if array[j] < pivot:

i = i + 1

swap array[i] and array[j]

swap array[i + 1] and array[high]

return i + 1

**CODE:**

#include <iostream>

using namespace std;

void swap(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

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

int pivot = array[high];

int i = low - 1;

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

if (array[j] < pivot) {

i++;

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

}

}

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

return i + 1;

}

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

if (low < high) {

int pivotIndex = partition(array, low, high);

quickSort(array, low, pivotIndex - 1);

quickSort(array, pivotIndex + 1, high);

}

}

int main() {

int size;

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

cin >> size;

int array[size];

cout << "Enter " << size << " elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

quickSort(array, 0, size - 1);

cout << "Sorted array:\n";

for (int i = 0; i < size; i++) {

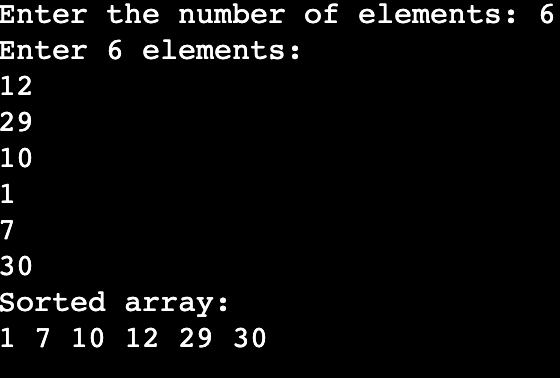
cout << array[i] << " ";

}

cout << endl;

return 0;

}



Name:- Udit Bansal

A023166923097

**EXPERIMENT-4**

Q) Write a program to implement merge sort

**PSEUDOCODE:**

function mergeSort(array, left, right):

if left < right:

mid = (left + right) / 2

mergeSort(array, left, mid)

mergeSort(array, mid + 1, right)

merge(array, left, mid, right)

function merge(array, left, mid, right):

create temporary arrays LeftArray and RightArray

copy data to LeftArray and RightArray

i = 0, j = 0, k = left

while i < length of LeftArray and j < length of RightArray:

if LeftArray[i] <= RightArray[j]:

array[k] = LeftArray[i]

i = i + 1

else:

array[k] = RightArray[j]

j = j + 1

k = k + 1

copy remaining elements of LeftArray, if any

copy remaining elements of RightArray, if any

**CODE:**

#include <iostream>

using namespace std;

void merge(int array[], int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int\* LeftArray = new int[n1];

int\* RightArray = new int[n2];

for (int i = 0; i < n1; i++)

LeftArray[i] = array[left + i];

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

RightArray[j] = array[mid + 1 + j];

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (LeftArray[i] <= RightArray[j]) {

array[k] = LeftArray[i];

i++;

} else {

array[k] = RightArray[j];

j++;

}

k++;

}

while (i < n1) {

array[k] = LeftArray[i];

i++;

k++;

}

while (j < n2) {

array[k] = RightArray[j];

j++;

k++;

}

delete[] LeftArray;

delete[] RightArray;

}

void mergeSort(int array[], int left, int right) {

if (left < right) {

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

mergeSort(array, left, mid);

mergeSort(array, mid + 1, right);

merge(array, left, mid, right);

}

}

int main() {

int size;

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

cin >> size;

int\* array = new int[size];

cout << "Enter " << size << " elements:\n";

for (int i = 0; i < size; i++) {

cin >> array[i];

}

mergeSort(array, 0, size - 1);

cout << "Sorted array:\n";

for (int i = 0; i < size; i++) {

cout << array[i] << " ";

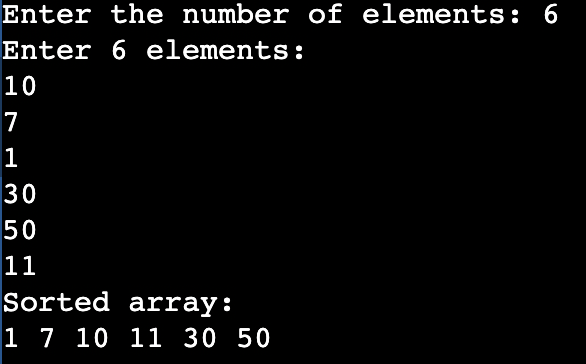
}

cout << endl;

delete[] array;

return 0;

}



Name:- Udit Bansal

A023166923097

**EXPERIMENT-5**

Q) Write a program to implement Strassens matrix for n elements

**PSEUDOCODE:**

Strassen(A, B, n)

1. If n = 1:

C[0][0] = A[0][0] \* B[0][0]

Return C

2. Else:

Divide A into 4 submatrices:

A11, A12, A21, A22

Divide B into 4 submatrices:

B11, B12, B21, B22

Compute 7 products (recursively):

M1 = Strassen(A11 + A22, B11 + B22)

M2 = Strassen(A21 + A22, B11)

M3 = Strassen(A11, B12 - B22)

M4 = Strassen(A22, B21 - B11)

M5 = Strassen(A11 + A12, B22)

M6 = Strassen(A21 - A11, B11 + B12)

M7 = Strassen(A12 - A22, B21 + B22)

Compute C’s submatrices:

C11 = M1 + M4 - M5 + M7

C12 = M3 + M5

C21 = M2 + M4

C22 = M1 - M2 + M3 + M6

Combine C11, C12, C21, C22 into C

3. Return C

**CODE:**

#include <iostream>

#include <vector>

using namespace std;

typedef vector<vector<int>> Matrix;

Matrix add(const Matrix &A, const Matrix &B) {

int n = A.size();

Matrix C(n, vector<int>(n, 0));

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

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

C[i][j] = A[i][j] + B[i][j];

return C;

}

Matrix subtract(const Matrix &A, const Matrix &B) {

int n = A.size();

Matrix C(n, vector<int>(n, 0));

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

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

C[i][j] = A[i][j] - B[i][j];

return C;

}

Matrix strassen(const Matrix &A, const Matrix &B) {

int n = A.size();

Matrix C(n, vector<int>(n, 0));

if (n == 1) {

C[0][0] = A[0][0] \* B[0][0];

return C;

}

int k = n / 2;

Matrix A11(k, vector<int>(k)), A12(k, vector<int>(k)),

A21(k, vector<int>(k)), A22(k, vector<int>(k));

Matrix B11(k, vector<int>(k)), B12(k, vector<int>(k)),

B21(k, vector<int>(k)), B22(k, vector<int>(k));

for (int i = 0; i < k; i++) {

for (int j = 0; j < k; j++) {

A11[i][j] = A[i][j];

A12[i][j] = A[i][j + k];

A21[i][j] = A[i + k][j];

A22[i][j] = A[i + k][j + k];

B11[i][j] = B[i][j];

B12[i][j] = B[i][j + k];

B21[i][j] = B[i + k][j];

B22[i][j] = B[i + k][j + k];

}

}

Matrix M1 = strassen(add(A11, A22), add(B11, B22));

Matrix M2 = strassen(add(A21, A22), B11);

Matrix M3 = strassen(A11, subtract(B12, B22));

Matrix M4 = strassen(A22, subtract(B21, B11));

Matrix M5 = strassen(add(A11, A12), B22);

Matrix M6 = strassen(subtract(A21, A11), add(B11, B12));

Matrix M7 = strassen(subtract(A12, A22), add(B21, B22));

Matrix C11 = add(subtract(add(M1, M4), M5), M7);

Matrix C12 = add(M3, M5);

Matrix C21 = add(M2, M4);

Matrix C22 = add(subtract(add(M1, M3), M2), M6);

for (int i = 0; i < k; i++) {

for (int j = 0; j < k; j++) {

C[i][j] = C11[i][j];

C[i][j + k] = C12[i][j];

C[i + k][j] = C21[i][j];

C[i + k][j + k] = C22[i][j];

}

}

return C;

}

void printMatrix(const Matrix &A) {

int n = A.size();

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

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

cout << A[i][j] << " ";

cout << endl;

}

}

int main() {

int n;

cout << "Enter matrix size (must be power of 2): ";

cin >> n;

Matrix A(n, vector<int>(n));

Matrix B(n, vector<int>(n));

cout << "Enter elements of Matrix A:\n";

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

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

cin >> A[i][j];

cout << "Enter elements of Matrix B:\n";

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

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

cin >> B[i][j];

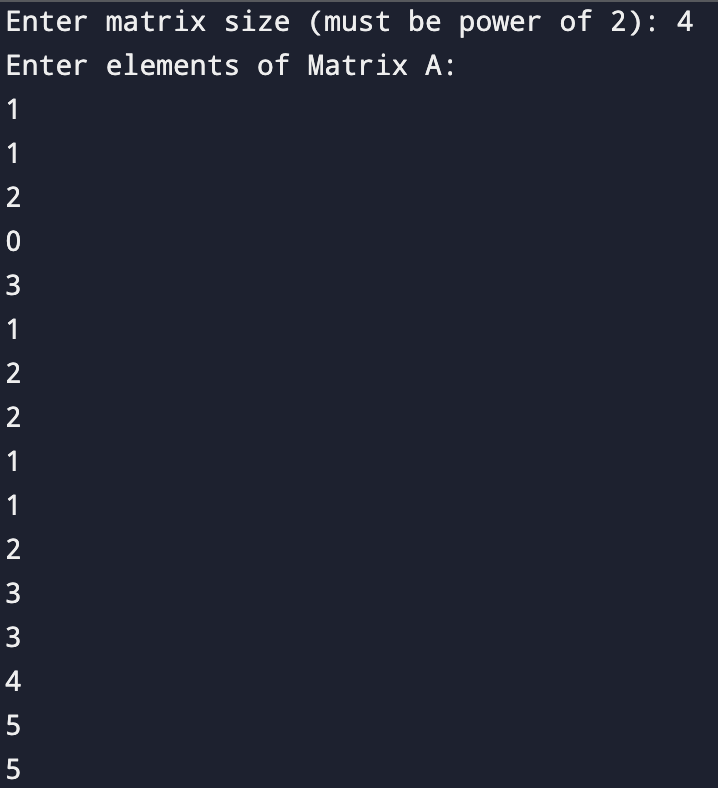
Matrix C = strassen(A, B);

cout << "\nResultant Matrix (A × B):\n";

printMatrix(C);

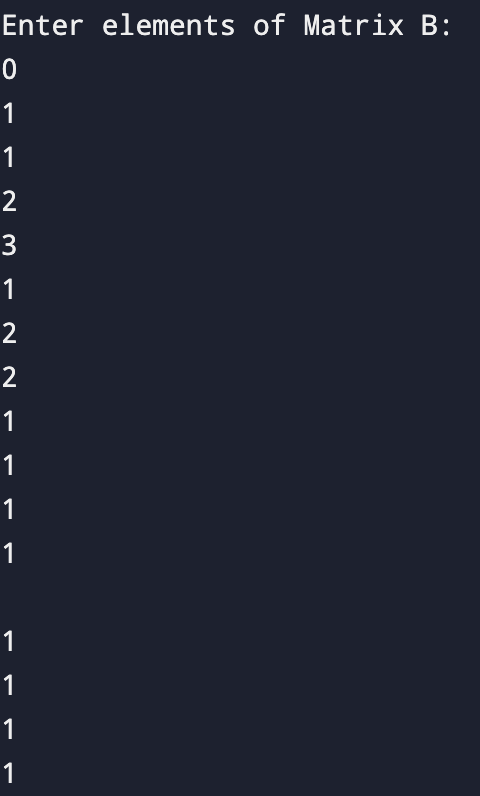
return 0;

}



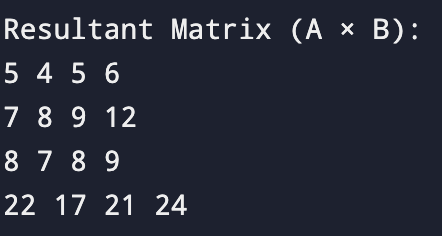
Name:- Udit Bansal

A023166923097



Name:- Udit Bansal

A023166923097



Name:- Udit Bansal

A023166923097