**DAY2 LAB**

**1 .Sum of rows and column in array**

**Code**:

#include <stdio.h>

int main() {

int rows, cols, i, j, sum = 0;

int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

printf("Array elements:\n");

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

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

}

printf("\n");

}

printf("\nSum of each row:\n");

for (i = 0; i < 3; i++) {

sum = 0;

for (j = 0; j < 3; j++) {

sum += arr[i][j];

}

printf("Sum of row %d: %d\n", i + 1, sum);

}

printf("\nSum of each column:\n");

for (i = 0; i < 3; i++) {

sum = 0;

for (j = 0; j < 3; j++) {

sum += arr[j][i];

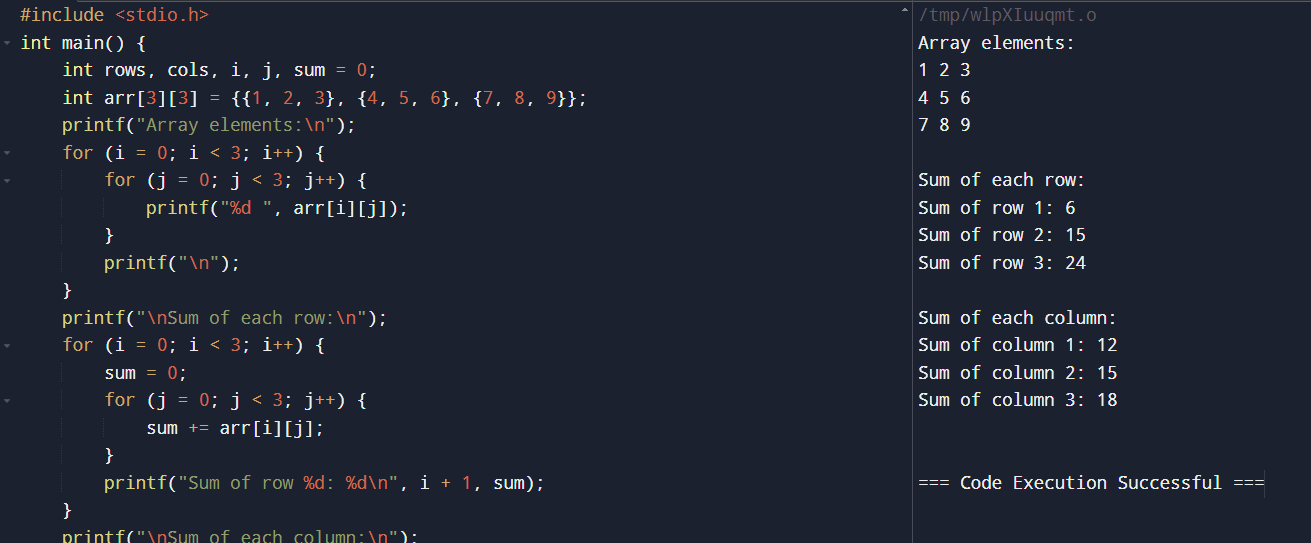
}

printf("Sum of column %d: %d\n", i + 1, sum);

}

}

**Output**:



**2. . Elements repeated twice – Array**

**Code**:

#include <stdio.h>

int main() {

int arr[] = {1, 2, 2, 3, 4, 4, 5};

int n = sizeof(arr) / sizeof(arr[0]);

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

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

if (arr[i] == arr[j]) {

printf("%d is repeated.\n", arr[i]);

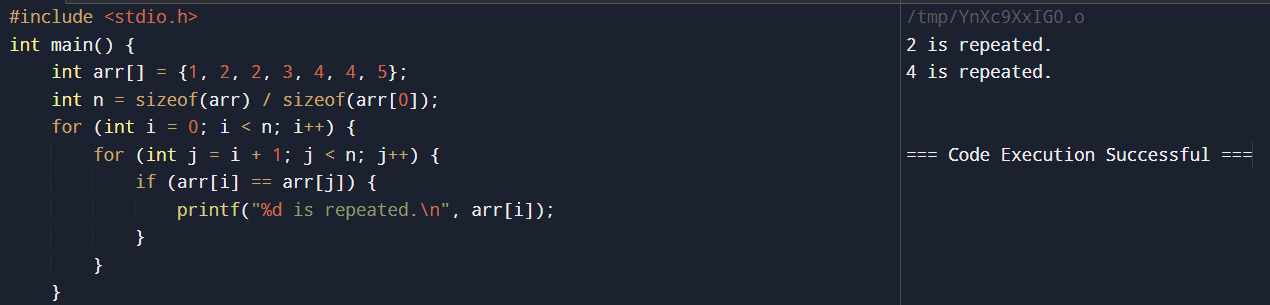
}

}

}

}

**Output**:



**3. matrix multiplication**

Code:

#include <stdio.h>

#define ROW1 2

#define COL1 2

#define ROW2 2

#define COL2 2

int main() {

int matrix1[ROW1][COL1] = {{1, 2}, {3, 4}};

int matrix2[ROW2][COL2] = {{1, 0}, {0, 1}};

int result[ROW1][COL2];

for (int i = 0; i < ROW1; i++) {

for (int j = 0; j < COL2; j++) {

result[i][j] = 0;

for (int k = 0; k < COL1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

printf("Result of Matrix Multiplication:\n");

for (int i = 0; i < ROW1; i++) {

for (int j = 0; j < COL2; j++) {

printf("%d ", result[i][j]);

}

printf("\n");

}

return 0;

}

**Output**:



**4. Write a C program to find Factorial of a given number without using Recursion**

Code:

#include <stdio.h>

int main() {

int fact=1,n,i;

scanf("%d",&n);

for(i=1;i<=n;i++){

fact=fact\*i;

}

printf("%d",fact);

}

**Output:**



**5.Fabanocci Series**

Code:

#include <stdio.h>

int main() {

int n, first = 0, second = 1, next, c;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

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

if (c <= 1)

next = c;

else {

next = first + second;

first = second;

second = next;

}

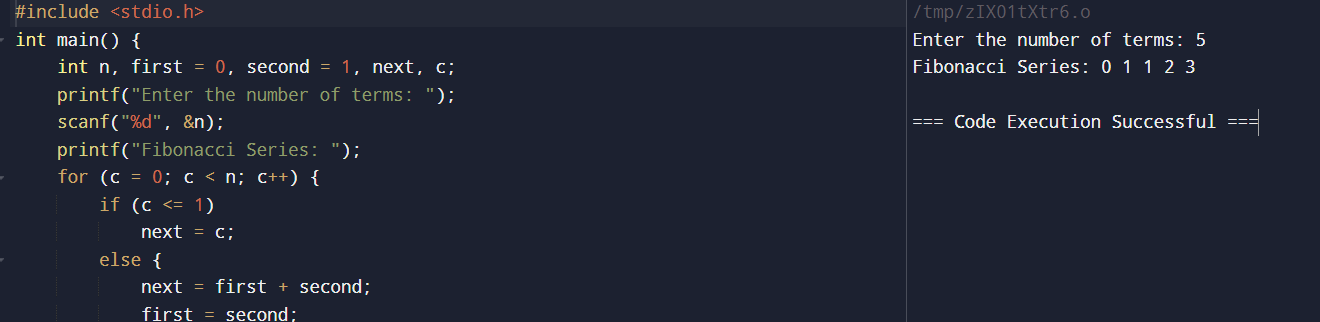
printf("%d ", next);

}

return 0;

}

**Output**:



**6. Write a C program to find Factorial of a given number using Recursion**

Code:

#include <stdio.h>

unsigned long long factorial(int n);

int main() {

int number;

printf("Enter a positive integer: ");

scanf("%d", &number);

if (number < 0)

printf("Error! Factorial of a negative number doesn't exist.");

else

printf("Factorial of %d = %llu", number, factorial(number));

return 0;

}

unsigned long long factorial(int n) {

if (n == 0)

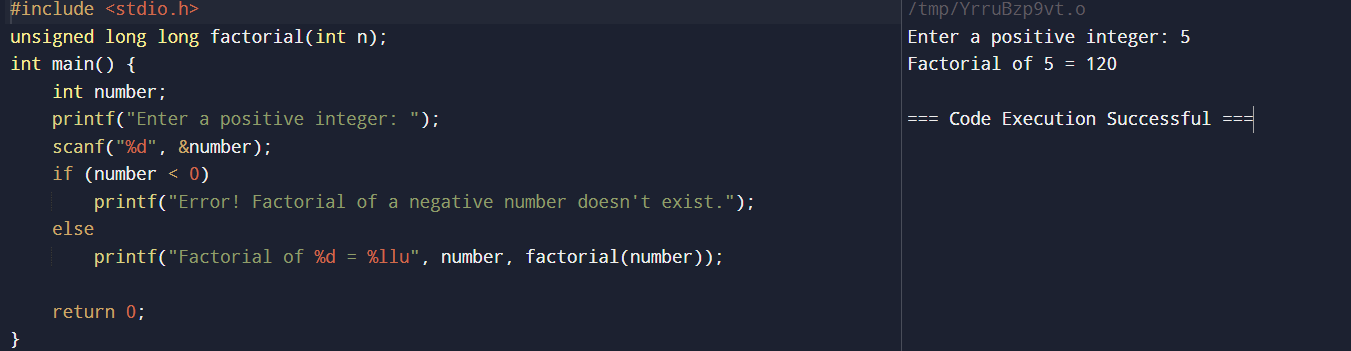
return 1;

else

return n \* factorial(n - 1);

}

**Output:**



**7.Write a C program to find Fibonacci series using Recursion**

Code:

#include <stdio.h>

int fibonacci(int n) {

if (n <= 1)

return n;

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main() {

int n, i;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

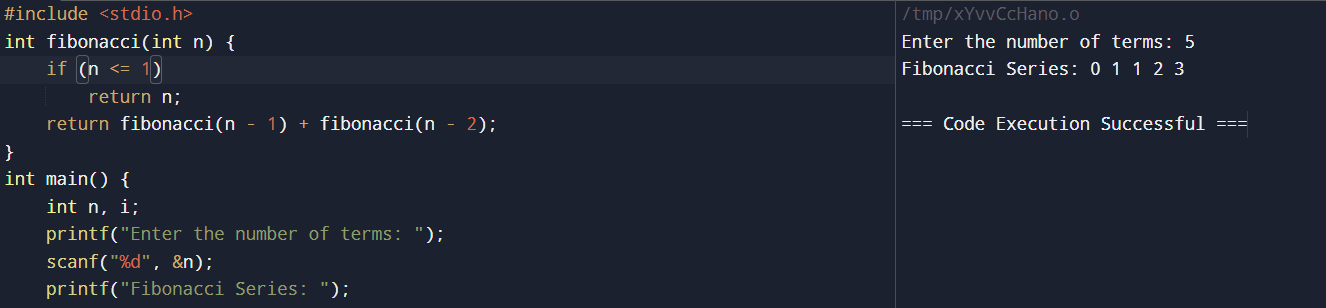
for (i = 0; i < n; i++) {

printf("%d ", fibonacci(i));

}

return 0;

}



**8.Write a C program to implement Array operations such as Insert, Delete and Display.**

Code:

#include <stdio.h>

#define MAX\_SIZE 100

void display(int arr[], int size) {

printf("Array elements: ");

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

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

}

printf("\n");

}

int insert(int arr[], int size, int element, int position) {

if (size >= MAX\_SIZE) {

printf("Array is full. Insertion not possible.\n");

return size;

}

for (int i = size - 1; i >= position; i--) {

arr[i + 1] = arr[i];

}

arr[position] = element;

return size + 1;

}

int delete(int arr[], int size, int position) {

if (size <= 0) {

printf("Array is empty. Deletion not possible.\n");

return size;

}

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

arr[i] = arr[i + 1];

}

return size - 1;

}

int main() {

int arr[MAX\_SIZE] = {1, 2, 3, 4, 5};

int size = 5;

display(arr, size);

size = insert(arr, size, 10, 2);

display(arr, size);

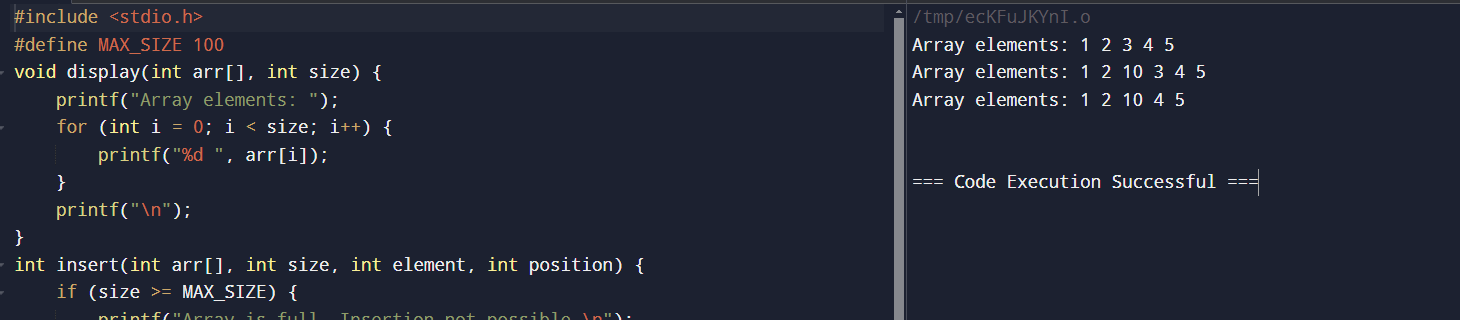
size = delete(arr, size, 3);

display(arr, size);

return 0;

}

**Output:**



**9. Write a C program to implement singly linked list**

Code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insertAtBeginning(struct Node\*\* head\_ref, int new\_data) {

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node\* node) {

while (node != NULL) {

printf("%d -> ", node->data);

node = node->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

insertAtBeginning(&head, 3);

insertAtBeginning(&head, 7);

insertAtBeginning(&head, 9);

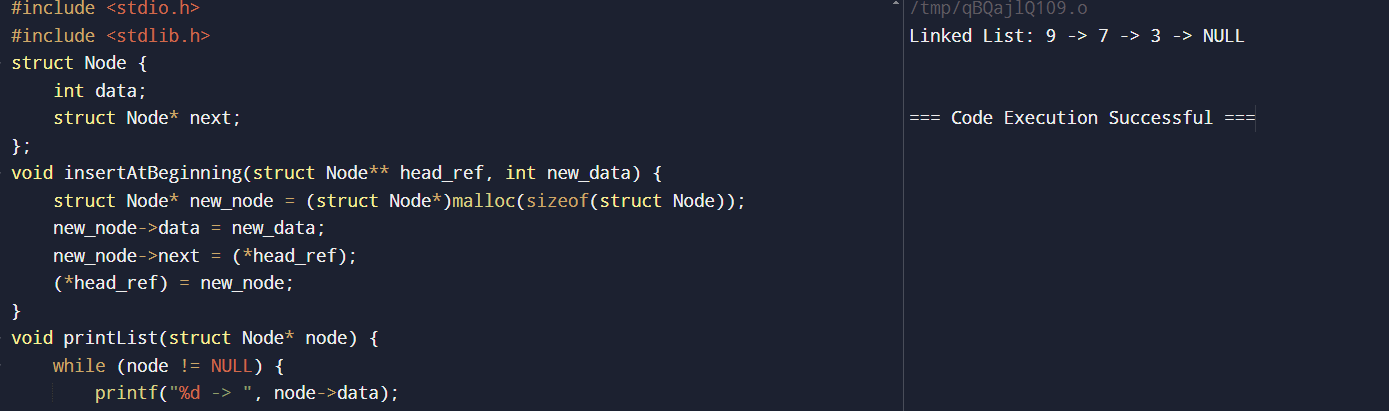
printf("Linked List: ");

printList(head);

return 0;

}

**Output:**



**10.Write a C program to implement doubly linked list**

Code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* prev;

struct Node\* next;

};

void insertAtBeginning(struct Node\*\* head\_ref, int new\_data);

void displayList(struct Node\* node);

int main() {

struct Node\* head = NULL;

insertAtBeginning(&head, 4);

insertAtBeginning(&head, 3);

insertAtBeginning(&head, 2);

insertAtBeginning(&head, 1);

printf("Doubly linked list: ");

displayList(head);

return 0;

}

void insertAtBeginning(struct Node\*\* head\_ref, int new\_data) {

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->prev = NULL;

new\_node->next = (\*head\_ref);

if ((\*head\_ref) != NULL) {

(\*head\_ref)->prev = new\_node;

}

(\*head\_ref) = new\_node;

}

void displayList(struct Node\* node) {

struct Node\* last;

while (node != NULL) {

printf("%d ", node->data);

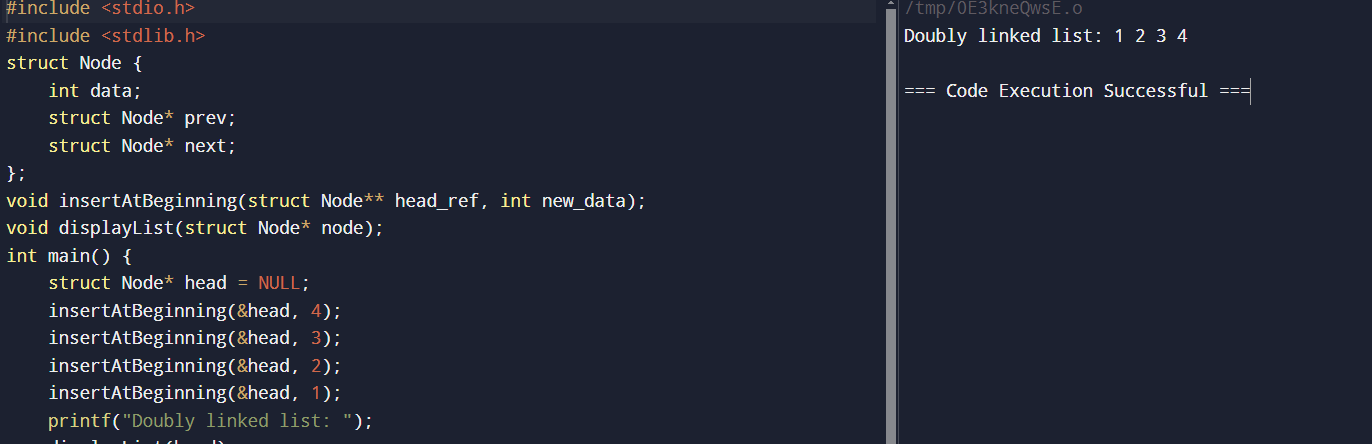
last = node;

node = node->next;

}

}

**Output:**



**11.Write a C program to implement circular linked list**

Code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insertAtBeginning(struct Node\*\* head\_ref, int new\_data) {

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* last = \*head\_ref;

new\_node->data = new\_data;

new\_node->next = \*head\_ref;

if (\*head\_ref != NULL) {

while (last->next != \*head\_ref)

last = last->next;

last->next = new\_node;

} else

new\_node->next = new\_node;

\*head\_ref = new\_node;

}

void displayList(struct Node\* head) {

struct Node\* temp = head;

if (head != NULL) {

do {

printf("%d ", temp->data);

temp = temp->next;

} while (temp != head);

}

}

int main() {

struct Node\* head = NULL;

insertAtBeginning(&head, 5);

insertAtBeginning(&head, 4);

insertAtBeginning(&head, 3);

insertAtBeginning(&head, 2);

insertAtBeginning(&head, 1);

printf("Circular Linked List: ");

displayList(head);

return 0;

}

**Output:**

