DAY-2-DA

1)sum of row and colum-array

```
#include <stdio.h>

#define ROWS 3

#define COLS 4

int main() {

    int matrix[ROWS][COLS] = {

        {1, 2, 3, 4},

        {5, 6, 7, 8},

        {9, 10, 11, 12}

    };

    int rowSums[ROWS] = {0};

    int colSums[COLS] = {0};

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

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

            rowSums[i] += matrix[i][j];

            colSums[j] += matrix[i][j];

        }

    }

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

        printf("Row %d: %d\n", i, rowSums[i]);

    }

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

        printf("Column %d: %d\n", j, colSums[j]);

    }
```

```
    return 0;
}
```

```
Output
/tmp/xTw7NKKvtP.o
Row 0: 10
Row 1: 26
Row 2: 42
Column 0: 15
Column 1: 18
Column 2: 21
Column 3: 24


=== Code Execution Successful ===
```

2)Elements repeated twice-array

```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 2, 3, 5, 6};
    int N = sizeof(arr) / sizeof(arr[0]) - 2;
    for (int i = 0; i < sizeof(arr) / sizeof(arr[0]); i++) {
        arr[arr[i] % (N + 1) - 1] += (N + 1);
        if ((arr[arr[i] % (N + 1) - 1] / (N + 1)) == 2) {
            printf("%d ", arr[i] % (N + 1));
        }
    }
    return 0;
}
```

```
Output
/tmp/gq4T7s6w2P.o
2 3

=== Code Execution Successful ===
```

3)write a c program to ferform matrix multiplication

```
#include <stdio.h>

#define MAX 10

int main() {

    int first[MAX][MAX], second[MAX][MAX], result[MAX][MAX];

    int rowFirst, colFirst, rowSecond, colSecond;

    scanf("%d %d", &rowFirst, &colFirst);

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

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

            scanf("%d", &first[i][j]);

        }

    }

    scanf("%d %d", &rowSecond, &colSecond);

    if (colFirst != rowSecond) {

        printf("Error: Number of columns in first matrix must be equal to number of rows in second
matrix.\n");

        return 1;

    }

    for (int i = 0; i < rowSecond; i++) {
```

```
        for (int j = 0; j < colSecond; j++) {
            scanf("%d", &second[i][j]);
        }
    }
    for (int i = 0; i < rowFirst; i++) {
        for (int j = 0; j < colSecond; j++) {2
            result[i][j] = 0;
        }
    }
    for (int i = 0; i < rowFirst; i++) {
        for (int j = 0; j < colSecond; j++) {
            for (int k = 0; k < colFirst; k++) {
                result[i][j] += first[i][k] * second[k][j];
            }
        }
    }
    printf("Resultant Matrix:\n");
    for (int i = 0; i < rowFirst; i++) {
        for (int j = 0; j < colSecond; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
Output
/tmp/rHN7I6puL4.o
2
2
2
2
2
2
2
2
2
2
22

2
Resultant Matrix:
8 8
8 8


=== Code Execution Successful ===
```

4)write a c program to find factorial of a given number without using recursion

```
#include <stdio.h>

int main() {

    int number;

    unsigned long long factorial = 1;

    scanf("%d", &number);

    if (number < 0) {

        return 1;

    }

    for (int i = 1; i <= number; i++) {
```

```
        factorial *= i;
    }
    printf("Factorial of %d = %llu\n", number, factorial);
    return 0;
}
```

```
Output
/tmp/410Gh9U1y2.o
6
Factorial of 6 = 720


=== Code Execution Successful ===
```

5)write a c program to find fibonacci series without using recurssion

```
#include <stdio.h>
int main() {
    int n, i;
    unsigned long long first = 0, second = 1, next;
    printf("Enter the number of terms in the Fibonacci series: ");
    scanf("%d", &n);
    if (n <= 0) {
        printf("Please enter a positive integer.\n");
    } else if (n == 1) {
        printf("Fibonacci Series: %llu\n", first);
    } else {
        printf("Fibonacci Series: %llu, %llu", first, second);
```

```
        for (i = 3; i <= n; i++) {
            next = first + second;
            printf(", %llu", next);
            first = second;
            second = next;
        }
        printf("\n");
    }
    return 0;
}
```

Output

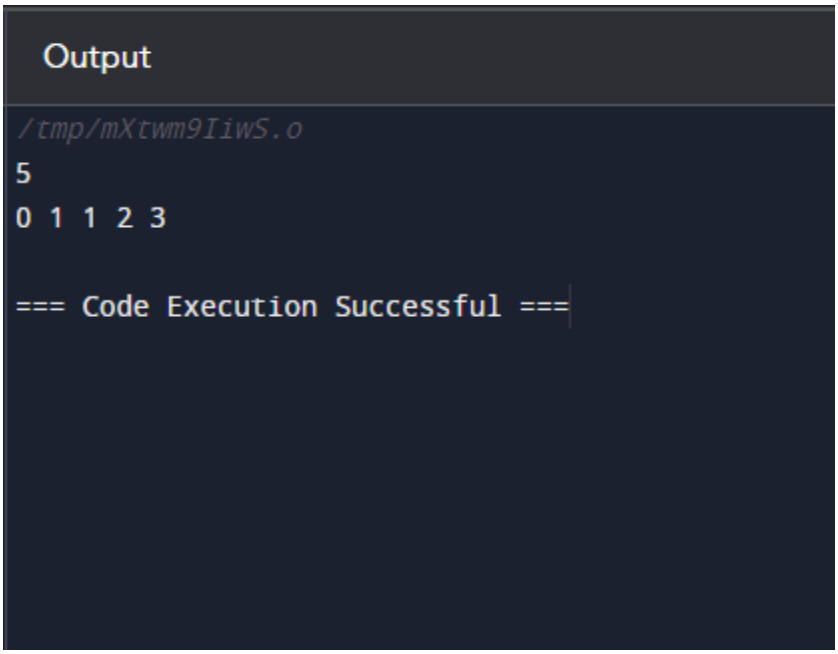
```
/tmp/BnIVeEYRMr.o
Enter the number of terms in the Fibonacci series: 5
Fibonacci Series: 0, 1, 1, 2, 3


=== Code Execution Successful ===
```

6)write a c program to find fibonacci series withusing recurssion

```
#include <stdio.h>
int main() {
    int n, i, t1 = 0, t2 = 1, nextTerm;
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        printf("%d ", t1);
```

```
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }
    return 0;
}
```



7)write a c program to find factorial of a given number withusing recursion

```
#include <stdio.h>
int main() {
    int number;
    unsigned long long factorial = 1;
    printf("Enter a non-negative integer: ");
    scanf("%d", &number);
    if (number < 0) {
```

```
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        int n = number;
        while (n > 1) {
            factorial *= n;
            n--;
        }
        printf("Factorial of %d is %llu\n", number, factorial);
    }
    return 0;
}
```

```
Output
/tmp/RZ06hoBssa.o
Enter a non-negative integer: 5
Factorial of 5 is 120


=== Code Execution Successful ===
```

8)write a c program to implement array operations such as insert,delete,and display

```
#include <stdio.h>
#define MAX_SIZE 100
int arr[MAX_SIZE];
int size = 0;
void insert(int position, int value) {
```

```
    if (size == MAX_SIZE) {
        printf("Error: Array is full.\n");
        return;
    }
    if (position < 0 || position > size) {
        printf("Error: Invalid position.\n");
        return;
    }
    for (int i = size; i > position; i--) {
        arr[i] = arr[i - 1];
    }
    arr[position] = value;
    size++;
}
void delete(int position) {
    if (size == 0) {
        printf("Error: Array is empty.\n");
        return;
    }
    if (position < 0 || position >= size) {
        printf("Error: Invalid position.\n");
        return;
    }
    for (int i = position; i < size - 1; i++) {
        arr[i] = arr[i + 1];
    }
    size--;
}
```

```
void display() {
	printf("Array elements:\n");
	for (int i = 0; i < size; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
}
int main() {
	insert(0, 10);
	insert(1, 20);
	insert(2, 30);
	display();
	insert(1, 15);
	display();
	delete(2);
	display();
	return 0;
}
```

```
Output
/tmp/AsvykFPxNS.o
Array elements:
10 20 30
Array elements:
10 15 20 30
Array elements:
10 15 30


=== Code Execution Successful ===
```

9)write a c program to implement singly linked list

```
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

	int data;

	struct Node* next;

} Node;

Node* createNode(int data) {

	Node* newNode = (Node*)malloc(sizeof(Node));

	newNode->data = data;

	newNode->next = NULL;

	return newNode;

}
```

```
Node* insertAtBeginning(Node* head, int data) {
	Node* newNode = createNode(data);
	newNode->next = head;
	return newNode;
}
Node* insertAtEnd(Node* head, int data) {
	Node* newNode = createNode(data);
	if (head == NULL) {
		return newNode;
	}
	Node* temp = head;
	while (temp->next != NULL) {
		temp = temp->next;
	}
	temp->next = newNode;
	return head;
}
void displayList(Node* head) {
	Node* temp = head;
	while (temp != NULL) {
		printf("%d ", temp->data);
		temp = temp->next;
	}
	printf("\n");
}
int main() {
	Node* head = NULL;
	head = insertAtBeginning(head, 3);
```

```
    head = insertAtBeginning(head, 2);
    head = insertAtBeginning(head, 1);
    printf("List after inserting at the beginning: ");
    displayList(head);
    head = insertAtEnd(head, 4);
    head = insertAtEnd(head, 5);
    printf("List after inserting at the end: ");
    displayList(head);
    return 0;
}
```

```
Output
/tmp/XiYpW9acvD.o
List after inserting at the beginning: 1 2 3
List after inserting at the end: 1 2 3 4 5


=== Code Execution Successful ===
```

10)write a c program to implement doubly linked list

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
```

```
int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* newNode = NULL;
    for (int i = 1; i <= 4; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = i * 10;
        newNode->next = NULL;
        newNode->prev = NULL;
        if (head == NULL) {
            head = newNode;
        } else {
            temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
            newNode->prev = temp;
        }
    }
    printf("Doubly Linked List: ");
    temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
    printf("Doubly Linked List in Reverse: ");
```

```
temp = head;
if (temp != NULL) {
	while (temp->next != NULL) {
		temp = temp->next;
	}
	while (temp != NULL) {
		printf("%d ", temp->data);
		temp = temp->prev;
	}
}
printf("\n");
if (head != NULL && head->next != NULL) {
	struct Node* delNode = head->next;
	if (delNode->next != NULL) {
		delNode->next->prev = delNode->prev;
	}
	delNode->prev->next = delNode->next;
	free(delNode);
}
printf("Doubly Linked List after deletion: ");
temp = head;
while (temp != NULL) {
	printf("%d ", temp->data);
	temp = temp->next;
}
printf("\n");
while (head != NULL) {
	struct Node* temp = head;
```

```
        head = head->next;
        free(temp);
    }
    return 0;
}
```

```
Output
/tmp/W1EAYHou01.o
Doubly Linked List: 10 20 30 40
Doubly Linked List in Reverse: 40 30 20 10
Doubly Linked List after deletion: 10 30 40


=== Code Execution Successful ===
```

11)write a c programto implement circular linked list

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    head = (struct Node*)malloc(sizeof(struct Node));
    head->data = 10;
```

```
head->next = head;
temp = (struct Node*)malloc(sizeof(struct Node));
temp->data = 20;
temp->next = head->next;
head->next = temp;
temp = (struct Node*)malloc(sizeof(struct Node));
temp->data = 30;
temp = head->next->next;
head->next->next = temp;
temp = head;
do {
	printf("%d -> ", temp->data);
	temp = temp->next;
} while (temp != head);
printf("(head)\n");
temp = head;
while (temp->next != head) {
	if (temp->next->data == 20) {
		struct Node* nodeToDelete = temp->next;
		temp->next = nodeToDelete->next;
		free(nodeToDelete);
		break;
	}
	temp = temp->next;
}
temp = head;
do {
	printf("%d -> ", temp->data);
```

```
        temp = temp->next;
    } while (temp != head);
    printf("(head)\n");

    return 0;
}
```

Output

```
/tmp/cuVFKcHF4o.o
10 -> 20 -> (head)
10 -> (head)


=== Code Execution Successful ===
```