S.No: 1	Exp. Name: ***Project Module***	**Date: 2024-05-09**

Aim:

Project Module

Source Code:

```
hello.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAME_LENGTH 100

// Structure to represent a node (user) in the graph
typedef struct User {
    char name[MAX_NAME_LENGTH];
    struct User* next;
} User;

// Structure to represent an adjacency list node
typedef struct AdjListNode {
    struct User* user;
    struct AdjListNode* next;
} AdjListNode;

// Structure to represent an adjacency list
typedef struct AdjList {
    AdjListNode* head;
} AdjList;

// Structure to represent a graph
typedef struct Graph {
    int numUsers;
    User* users;
    AdjList* array;
} Graph;

// Function to create a new user node
User* createUser(char* name) {
    User* newUser = (User*)malloc(sizeof(User));
    strncpy(newUser->name, name, MAX_NAME_LENGTH);
    newUser->next = NULL;
    return newUser;
}

// Function to create a new adjacency list node
AdjListNode* createAdjListNode(User* user) {
    AdjListNode* newNode =
(AdjListNode*)malloc(sizeof(AdjListNode));
    newNode->user = user;
    newNode->next = NULL;
    return newNode;
}
```

```c
// Function to create a graph
Graph* createGraph() {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->numUsers = 0;
    graph->users = NULL;
    graph->array = NULL;
    return graph;
}

// Function to add a user to the graph
void addUser(Graph* graph, char* name) {
    // Create a new user node
    User* newUser = createUser(name);

    // Add the new user to the user list
    newUser->next = graph->users;
    graph->users = newUser;
    graph->numUsers++;

    // Reallocate memory for the adjacency list array
    graph->array = (AdjList*)realloc(graph->array, graph->numUsers
* sizeof(AdjList));
    graph->array[graph->numUsers - 1].head = NULL;
}

// Function to get a user by name
User* getUser(Graph* graph, char* name) {
    User* temp = graph->users;
    while (temp) {
        if (strncmp(temp->name, name, MAX_NAME_LENGTH) == 0) {
            return temp;
        }
        temp = temp->next;
    }
    return NULL;
}

// Function to add an edge (friendship) between two users
void addEdge(Graph* graph, char* srcName, char* destName) {
    User* srcUser = getUser(graph, srcName);
    User* destUser = getUser(graph, destName);
    if (srcUser == NULL || destUser == NULL) {
        printf("One or both users not found.\n");
        return;
    }
    int srcIndex = -1, destIndex = -1;
    User* temp = graph->users;
    for (int i = 0; i < graph->numUsers; ++i) {
        if (temp == srcUser) {
            srcIndex = i;
        }
        if (temp == destUser) {
            destIndex = i;
        }
        temp = temp->next;
    }

    // Add an edge from src to dest
    AdjListNode* newNode = createAdjListNode(destUser);
    newNode->next = graph->array[srcIndex].head;
    graph->array[srcIndex].head = newNode;

    // Since the graph is undirected, add an edge from dest to src
as well
    newNode = createAdjListNode(srcUser);
    newNode->next = graph->array[destIndex].head;
    graph->array[destIndex].head = newNode;
}

// Function to print the adjacency list representation of the
graph
void printGraph(Graph* graph) {
    User* tempUser = graph->users;
    int i = 0;
    while (tempUser) {
        printf("User %s: ", tempUser->name);
        AdjListNode* tempAdj = graph->array[i].head;
        while (tempAdj) {
            printf("%s ", tempAdj->user->name);
            tempAdj = tempAdj->next;
        }
        printf("\n");
        tempUser = tempUser->next;
        i++;
    }
}

// Function to free the memory allocated for the graph
void freeGraph(Graph* graph) {
    for (int i = 0; i < graph->numUsers; ++i) {
        AdjListNode* tempAdj = graph->array[i].head;
        while (tempAdj) {
            AdjListNode* next = tempAdj->next;
            free(tempAdj);
            tempAdj = next;
        }
    }

    free(graph->array);

    User* tempUser = graph->users;
    while (tempUser) {
        User* next = tempUser->next;
        free(tempUser);
        tempUser = next;
    }

    free(graph);
}

int main() {
    Graph* graph = createGraph();

    addUser(graph, "Alice");
    addUser(graph, "Bob");
    addUser(graph, "Charlie");
    addUser(graph, "David");

    addEdge(graph, "Alice", "Bob");
    addEdge(graph, "Alice", "Charlie");
    addEdge(graph, "Bob", "David");
    addEdge(graph, "Charlie", "David");

    printGraph(graph);

    // Free allocated memory
    freeGraph(graph);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Hello World
```