```c
// 171805024 Nagihan BAZ
#include <stdio.h>
#include <stdlib.h>


struct graph {
    char vertice[10];
    char edge[20];
    char endpoint[10];
    int number;
    struct graph* nextPtr;
};

typedef struct graph graph;
typedef graph* graphPtr;


void insert(graphPtr* sPtr, char vertice[10], char edge[20]);
void insert(graphPtr* sPtr, char endpoint[10]);
void printList(graphPtr currentPtr);
void instructions(void);

int main(void)
{
    graphPtr startPtr = NULL;
    unsigned int choice;
    char item;

    instructions();
    printf("%s", "Please enter 1 to read the information of an undirected graph.");
    scanf("%u", &choice);
    printf("%s", "Please enter 2 to find and print the adjacency matrix of the graph.");
    scanf("%u", &choice);
    printf("%s", "Please enter 3 to find and print the incidence matrix of the graph.");
    scanf("%u", &choice);
    printf("%s", "Please enter 4 to find and print the degree of the graph (largest vertex
degree).");
    scanf("%u", &choice);
    printf("%s", "Please enter 5 to exit the program.");
    scanf("%u", &choice);



    while (choice != 5) {

        switch (choice) {
        case 1:
            printf("%s", "Enter to read the information of an undirected graph:");
            scanf("\n%c", &item);
            printList(startPtr);

            break;
        case 2:
            printf("%s", "Enter to find and print the adjacency matrix of the graph:");
            scanf("\n%c", &item);

            int main();
            {
```

```
            int Array[10][10], vertice[10], RC, CC;
            printf("Please the number of rows \n"); scanf("%d", &vertice[10]);
            printf("Please the number of columns \n"); scanf("%d", &vertice[10]);
            for (RC = 0; RC < vertice[10]; RC++)
                for (CC = 0; CC < vertice[10]; CC++)
                {
                    printf("Plz enter an element for row= %d, column= %d:", RC, CC);
                    scanf("%d", &Array[RC][CC]);
                }
            for (RC = 0; RC < vertice[10]; RC++)
                for (CC = 0; CC < vertice[10]; CC++)
                {
                    printf("%3d ", Array[RC][CC]);
                    if (CC == vertice[10] - 1) printf("\n");
                }
            system("pause");
            return 0;
        }

        printList(startPtr);

        break;
    case 3:
        printf("%s", "Enter to find and print the incidence matrix of the graph:");
        scanf("\n%c", &item);


        int main();
        {
            int Array[20][20], vertice[10], edge[20], RC, CC;
            printf("Please the number of rows \n"); scanf("%d", &vertice[10]);
            printf("Please the number of columns \n"); scanf("%d", &edge[20]);
            for (RC = 0; RC < vertice[10]; RC++)
                for (CC = 0; CC < edge[20]; CC++)
                {
                    printf("Plz enter an element for row= %d, column= %d:", RC, CC);
                    scanf("%d", &Array[RC][CC]);
                }
            for (RC = 0; RC < vertice[10]; RC++)
                for (CC = 0; CC < edge[20]; CC++)
                {
                    printf("%3d ", Array[RC][CC]);
                    if (CC == edge[20] - 1) printf("\n");
                }
            system("pause");
            return 0;
        }
        printList(startPtr);
        break;
    case 4:
        printf("%s", "Enter to find and print the degree of the graph (largest vertex
degree):");
        scanf("\n%c", &item);


        int main();
        {
            printf("Find and print the degree of the graph (largest vertex degree)");
```

```
                int vertex[10], i, largest;
                vertex[1] = largest;
                for (i = 1; i <= 10; i++) {
                    printf("%d", "Enter vertex degree:", i);
                    scanf("%d", &vertex[i]);
                    if (vertex[i] > largest);
                    largest = vertex[i];

                }
                printf("Largest vertex degree is: %d", largest);
                getch();
            }

            printList(startPtr);

            break;
        default:
            puts("Invalid choice.\n");
            instructions();
            break;
        }

        printf("%s", "Enter your choice.");
        scanf("%u", &choice);
    }

    puts("End of run.");
}

void instructions(void)
{
    puts("Enter your choice:\n"
        "   1 to read the information of an undirected graph.\n"
        "   2 to find and print the adjacency matrix of the graph.\n"
        "   3 to find and print the incidence matrix of the graph.\n"
        "   4 to find and print the degree of the graph (largest vertex degree).\n"
        "   5 to end.\n");
}

void insert(graphPtr* sPtr, char vertice[10])
{
    graphPtr newPtr;
    graphPtr previousPtr;
    graphPtr currentPtr;

    newPtr = malloc(sizeof(graph));

    if (newPtr != NULL) {
        newPtr->vertice[10] = vertice[10];
        newPtr->nextPtr = NULL;

        previousPtr = NULL;
        currentPtr = *sPtr;


        while (currentPtr != NULL && vertice > currentPtr->vertice[10]) {
            previousPtr = currentPtr;
            currentPtr = currentPtr->nextPtr;
```

```
        }


        if (previousPtr == NULL) {
            newPtr->nextPtr = *sPtr;
            *sPtr = newPtr;
        }
        else {
            previousPtr->nextPtr = newPtr;
            newPtr->nextPtr = currentPtr;
        }
    }
    else {
        printf("%c not inserted. No memory available.\n", vertice);
    }
}
void insert(graphPtr* sPtr, char edge[20])
{
    graphPtr newPtr;
    graphPtr previousPtr;
    graphPtr currentPtr;

    newPtr = malloc(sizeof(graph));

    if (newPtr != NULL) {
        newPtr->edge[20] = edge[20];
        newPtr->nextPtr = NULL;

        previousPtr = NULL;
        currentPtr = *sPtr;


        while (currentPtr != NULL && edge > currentPtr->edge[20]) {
            previousPtr = currentPtr;
            currentPtr = currentPtr->nextPtr;
        }


        if (previousPtr == NULL) {
            newPtr->nextPtr = *sPtr;
            *sPtr = newPtr;
        }
        else {
            previousPtr->nextPtr = newPtr;
            newPtr->nextPtr = currentPtr;
        }
    }
    else {
        printf("%c not inserted. No memory available.\n", edge);
    }
    void insert(graphPtr * sPtr, char endpoint[10]);
    int endpoint[10];
    {
        graphPtr newPtr;
        graphPtr previousPtr;
        graphPtr currentPtr;

        newPtr = malloc(sizeof(graph));
```

```
        if (newPtr != NULL) {
            newPtr->endpoint[10] = endpoint[10];
            newPtr->nextPtr = NULL;

            previousPtr = NULL;
            currentPtr = *sPtr;


            while (currentPtr != NULL && endpoint > currentPtr->endpoint[10]) {
                previousPtr = currentPtr;
                currentPtr = currentPtr->nextPtr;
            }


            if (previousPtr == NULL) {
                newPtr->nextPtr = *sPtr;
                *sPtr = newPtr;
            }
            else {
                previousPtr->nextPtr = newPtr;
                newPtr->nextPtr = currentPtr;
            }
        }
        else {
            printf("%c not inserted. No memory available.\n", endpoint);
        }
    }


}
```