```c
//WAP to create graph using linked list

#include <stdio.h>
#include <stdlib.h>
#define new_node (struct node*)malloc(sizeof(struct node))

struct node {
    int vertex;
    struct node *next;
};
void main() {
    int ch_2105734;
    do {
        printf(
                "\n A Program to represent a Graph by using an Linked List \n ");
        printf("\n 1. Directed Graph ");
        printf("\n 2. Un-Directed Graph ");
        printf("\n 3. Exit ");
        printf("\n\n Select a proper option : ");
        scanf("%d", &ch_2105734);
        switch (ch_2105734) {
            case 1:
                dirgraph_2105734();
                break;
            case 2:
                undirgraph_2105734();
                break;
            case 3:
                exit(0);
        }
    } while (1);
}
int dirgraph_2105734() {
    struct node *neighbour_2105734[10], *p;
    int n;
    int indeg_2105734, outdeg_2105734, i, j;
    printf("\n How Many Vertices ? : ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
        neighbour_2105734[i] = NULL;
    read_graph(neighbour_2105734, n);
    printf("\n Vertex \t In_Degree \t Out_Degree \t Total_Degree ");
    for (i = 1; i <= n; i++) {
        indeg_2105734 = outdeg_2105734 = 0;
        p = neighbour_2105734[i];
        while (p != NULL) {
            outdeg_2105734++;
            p = p -> next;
        }
        for (j = 1; j <= n; j++) {
            p = neighbour_2105734[j];
            while (p != NULL) {
                if (p -> vertex == i)
                    indeg_2105734++;
                p = p -> next;
                }
        }
        printf("\n\n %5d\t\t\t%d\t\t%d\t\t%d\n\n", i, indeg_2105734, outdeg_2105734,
                indeg_2105734 + outdeg_2105734);
    }
    return;
```

```
}
int undirgraph_2105734() {
    struct node *neighbour_2105734[10], *p;
    int deg_2105734, i, j, n;
    printf("\n How Many Vertices ? : ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
        neighbour_2105734[i] = NULL;
    read_graph(neighbour_2105734, n);
    printf("\n Vertex \t Degree ");
    for (i = 1; i <= n; i++) {
        deg_2105734 = 0;
        p = neighbour_2105734[i];
        while (p != NULL) {
            deg_2105734++;
            p = p -> next;
        }
        printf("\n\n %5d \t\t %d\n\n", i, deg_2105734);
    }
    return;
}
int read_graph(struct node *neighbour_2105734[10], int n) {
    int i, j;
    char reply_2105734;
    struct node *p, *c;
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            if (i == j)
                continue;
            printf("\n Vertices %d & %d are Adjacent ? (Y/N) :", i, j);
            scanf("%c", &reply_2105734);
            if (reply_2105734 == 'y' || reply_2105734 == 'Y') {
                c = new_node;
                c -> vertex = j;
                c -> next = NULL;
                if (neighbour_2105734[i] == NULL)
                  neighbour_2105734[i] = c;
                else {
                    p = neighbour_2105734[i];
                    while (p -> next != NULL)
                        p = p -> next;
                    p -> next = c;
                }
            }
        }
    }
    return;
}
```

OUTPUT

A Program to represent a Graph by using an Linked List

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : 1

How Many Vertices ? : 2

Vertices 1 & 2 are Adjacent ? (Y/N) :
Vertices 2 & 1 are Adjacent ? (Y/N) :Y

Vertex	In_Degree	Out_Degree	Total_Degree
1	1	0	1
2	0	1	1

A Program to represent a Graph by using an Linked List

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option :

RISHIKESH
2105734
CSE-32