```
#include <iostream>
#include <cstdlib>
using namespace std;
class Queue{
    int front, rear;
    int capacity;
    int *q;
public:
    Queue(int size) : front(-1), rear(-1), capacity(size){
        q = (int *)malloc(capacity * sizeof(int)); }
    ~Queue(){
        free(q); }
    bool isEmpty() { return (front == -1 && rear == -1); }
    bool isFull() { return (rear == capacity - 1); }
    void enqueue(int value){
        if (isFull())
            cout << "Queue Overflow" << endl;
        else{
            if (isEmpty())
                front = 0;
            q[++rear] = value;}}
    int dequeue(){
        if (isEmpty()){
            cout << "Queue Underflow" << endl;
            return -1;}
        else{
            int value = q[front];
            if (front == rear)
                front = rear = -1;
            else
                front++;
            return value;}}};
class Stack{
    int top;
    int capacity;
    int *s;
public:
    Stack(int size) : top(-1), capacity(size){
        s = (int *)malloc(capacity * sizeof(int)); }
    ~Stack(){
        free(s); }
    bool isEmpty() { return (top == -1); }
    bool isFull() { return (top == capacity - 1); }
    void push(int value){
        if (isFull())
            cout << "Stack Overflow" << endl;
        else
            s[++top] = value;}
    int pop(){
        if (isEmpty()){
            cout << "Stack Underflow" << endl;
            return -1;}
        else
            return s[top--]; }};
class gnode{
public:
    int vertex;
    gnode *next;
    friend class Graph;};
class Graph{
private:
    gnode *head[20];
    int n;
    bool visited[200];
public:
    Graph(){
```

[redacted]

```
cout << "Enter the number of users: ";
        cin >> n;
        for (int i = 0; i < n; i++){
            head[i] = NULL;}}
    void create(){
        for (int i = 0; i < n; i++){
            gnode *temp = head[i];
            char ans;
            do{
                int v;
                cout << "Enter adjacent vertex for user " << i << ": ";
                cin >> v;
                if (v == i)
                    cout << "Self loops are not allowed." << endl;
                else{
                    gnode *curr = (gnode *)malloc(sizeof(gnode));
                    curr->vertex = v;
                    curr->next = NULL;
                    if (head[i] == NULL)
                        head[i] = curr;
                    else
                        temp->next = curr;
                    temp = curr;}
                cout << "Do you want to add another adjacent vertex for user " << i
<< "? (y/n): ";
                cin >> ans;} while (ans == 'y' || ans == 'Y');}
        cout << "Graph created successfully." << endl;}
    void display(){
        for (int i = 0; i < n; i++){
            gnode *temp = head[i];
            cout << i << " -> ";
            while (temp != NULL){
                cout << temp->vertex << " ";
                temp = temp->next;}
            cout << endl;}}
    void DFSr(){
        int v;
        cout << "Enter starting vertex: ";
        cin >> v;
        DFSr(v);}
void DFSr(int v){
        cout << v << " ";
        visited[v] = true;
        gnode *temp = head[v];
        while (temp != NULL){
            if (!visited[temp->vertex])
                DFSr(temp->vertex);
            temp = temp->next;}}
void DFSnr(){
        Stack s(n);
        int v;
        cout << "Enter starting vertex: ";
        cin >> v;
        for (int i = 0; i < n; i++){
            visited[i] = false;}
        s.push(v);
        visited[v] = true;
        while (!s.isEmpty()){
            v = s.pop();
            cout << " " << v;
            gnode *temp = head[v];
            while (temp != NULL){
                if (!visited[temp->vertex]){
                    s.push(temp->vertex);
                    visited[temp->vertex] = true;}
                temp = temp->next;}}}
```

[redacted]

```
    void BFS(int v){
        for (int i = 0; i < n; i++)
            visited[i] = 0;
        Queue q(n);
        q.enqueue(v);
        visited[v] = 1;
        while (!q.isEmpty()){
            v = q.dequeue();
            cout << v << " ";
            gnode *temp = head[v];
            while (temp != NULL){
                int w = temp->vertex;
                if (!visited[w]){
                    q.enqueue(w);
                    visited[w] = 1;}
                temp = temp->next;}}}};
int main()
{
    int v;
    Graph g;
    g.create();
    g.display();
    cout << "DFS Traversal (Recursive): \n";
    g.DFSr();
    cout << endl;
    cout << "DFS Traversal (Non-Recursive): \n";
    g.DFSnr();
    cout << endl;
    cout << "BFS Traversal: \n";
    cout << "Enter starting vertex: ";
    cin >> v;
    g.BFS(v);
    return 0;
}
```

[redacted]