DATE / /
PAGE NO. Usha Gold

BSE

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

#define MAX_NODES 100

struct Node {
    int data;
    struct Node* next;
};

struct Graph {
    int numNodes;
    struct Node* adjLists[MAX_NODES];
    int visited[MAX_NODES];
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)
        malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
```

```
struct Graph* createGraph(int n){
    struct Graph* graph = (struct Graph
        *) malloc (sizeof(struct
            Graph));
    graph -> numNodes = n;
    for(int i=0; i<n; i++){
        graph -> adjLists[i] = Nu
        graph -> visited[i] = 0;
    }
    return graph;
}

void addEdge(struct Graph* graph, int
        src, int dest){
    struct Node* newNode = (createNode(
        dest);
    newNode -> next = graph -> adjLists[s
        c];
    graph -> adjLists[src] = newNode;

    newNode = createNode(src);
    newNode -> next = graph -> adjLists
        [dest];
    graph -> adjLists[dest] = newNode;
}
```

```
void BFS(struct Graph, int StartNode){
    int queue[MAX_NODES];
    int front=0, rear=0;

    graph->visited[startNode]=1;
    queue[rear++]=StartNode;

    while(front<rear){
        int current=queue[front++];
        printf("%d", current);

        struct Node* temp = graph->
                adjlists[current];
        while(temp){
            int adjNode = temp->data;
            if(!graph->visited[
                    adjNode]){
                graph->visited[adjNode]=1
                queue[rear++]=adjNode;
            }
            temp=temp->next;
        }
    }
}
```

```
int main(){
    int numNodes;
    printf("Enter the number of
                nodes:");
    scanf("%d", &numNodes);

    struct Graph* graph = CreateGra
                ph(numNo
                des);

    int numEdges;
    printf("Enter the number of
                edges:");
    scanf("%d", &numEdges);

    for(int i=0; i<numEdges; i++){
        int src, dest;
        printf("Enter edge %d (source
                destination):", i+1);

        scanf("%d %d", &src, &dest);
        addEdge(graph, src, dest);
    }

    int startNode;
    printf("Enter the starting node for
```

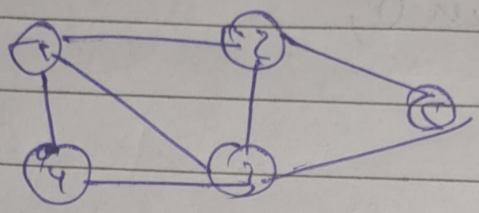
BFS traversal :");
scanf("%d", &startNode);
printf("BFS traversal starting from node %d:", startNode);
BFS(graph, startNode);

return 0;
}

Output:

Enter the number of node
Enter the number of edges
Enter edge 1 (source destination): 1
2
Enter edge 2 (Source destination): 1
3
Enter edge 3 (source destination): 1
4
Enter edge 4 (source destination): 2
5
Enter edge 5 (source destination): 2
6
Enter edge 6 (source destination): 3
4

Enter edge 7 (source destination):
5
Enter starting node for BFS traversal
Sol 1
BFS traversal starting from node
1 4 3 2 5



Delete a Node in BST

```
struct TreeNode * inorderSuccessor(
  struct TreeNode* root){
  struct TreeNode * curr = root;
  while(curr && curr->left != NULL)
      curr = curr->left;
  return curr;
}
```

```
struct TreeNode * deleteNode(struct Tree
  node * root, int key){
  if(root == NULL) return root;
  if(key < (root->val)) root->left =
      deleteNode(root->left, key);
  else if(k > (root->val))
      root->right = deleteNode(root->
          right, key);
  else{
      if(root->left == NULL){
          struct TreeNode * temp =
              root->left;
          free(root);
          return temp;
      }
```

11/2/24

```
else {
    struct TreeNode* temp =
        inorderSuccessor(root->right);
    root->val = temp->val;
    root->right = deleteNode
        (root->right, temp->
        val);
}
}
return root;
}
```