main.c

Run

```c
#192210211
#include <stdio.h>
#include <stdlib.h>
#define SIZE 40
struct queue {
  int items[SIZE];
  int front;
  int rear;};
struct queue* createQueue();
void enqueue(struct queue* q, int);
int dequeue(struct queue* q);
void display(struct queue* q);
int isEmpty(struct queue* q);
void printQueue(struct queue* q);
struct node {
  int vertex;
  struct node* next;};
struct node* createNode(int);
struct Graph {
  int numVertices;
  struct node** adjLists;
  int* visited;};
void bfs(struct Graph* graph, int startVertex) {
  struct queue* q = createQueue();
  graph->visited[startVertex] = 1;
  enqueue(q, startVertex);
  while (!isEmpty(q)) {
    printQueue(q);
    int currentVertex = dequeue(q);
    printf("Visited %d\n", currentVertex);
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Queue contains
0 Resetting queue Visited 0

Queue contains
2 1 Visited 2

Queue contains
1 4 Visited 1

Queue contains
4 3 Visited 4

Queue contains
3 Resetting queue Visited 3
```

main.c

Run

```
31      struct node* temp = graph->adjLists[currentVertex];
32      while (temp) {
33        int adjVertex = temp->vertex;
34        if (graph->visited[adjVertex] == 0) {
35          graph->visited[adjVertex] = 1;
36          enqueue(q, adjVertex);}
37        temp = temp->next;}}}
38  struct node* createNode(int v) {
39    struct node* newNode = malloc(sizeof(struct node));
40    newNode->vertex = v;
41    newNode->next = NULL;
42    return newNode;}
43  struct Graph* createGraph(int vertices) {
44    struct Graph* graph = malloc(sizeof(struct Graph));
45    graph->numVertices = vertices;
46    graph->adjLists = malloc(vertices * sizeof(struct node*));
47    graph->visited = malloc(vertices * sizeof(int));
48    int i;
49    for (i = 0; i < vertices; i++) {
50      graph->adjLists[i] = NULL;
51      graph->visited[i] = 0;}
52    return graph;}
53  void addEdge(struct Graph* graph, int src, int dest) {
54    struct node* newNode = createNode(dest);
55    newNode->next = graph->adjLists[src];
56    graph->adjLists[src] = newNode;
57    newNode = createNode(src);
58    newNode->next = graph->adjLists[dest];
59    graph->adjLists[dest] = newNode;}
60  struct queue* createQueue() {
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Queue contains
0 Resetting queue Visited 0

Queue contains
2 1 Visited 2

Queue contains
1 4 Visited 1

Queue contains
4 3 Visited 4

Queue contains
3 Resetting queue Visited 3
```

main.c

Run

```
  graph->adjLists[dest] = newNode;}
struct queue* createQueue() {
  struct queue* q = malloc(sizeof(struct queue));
  q->front = -1;
  q->rear = -1;
  return q;}
int isEmpty(struct queue* q) {
  if (q->rear == -1)
    return 1;
  else
    return 0;}
void enqueue(struct queue* q, int value) {
  if (q->rear == SIZE - 1)
    printf("\nQueue is Full!!");
  else {
    if (q->front == -1)
      q->front = 0;
    q->rear++;
    q->items[q->rear] = value;}}
int dequeue(struct queue* q) {
  int item;
  if (isEmpty(q)) {
    printf("Queue is empty");
    item = -1;
  } else {
    item = q->items[q->front];
    q->front++;
    if (q->front > q->rear) {
      printf("Resetting queue ");
      q->front = q->rear = -1;}}
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Queue contains
0 Resetting queue Visited 0

Queue contains
2 1 Visited 2

Queue contains
1 4 Visited 1

Queue contains
4 3 Visited 4

Queue contains
3 Resetting queue Visited 3
```

main.c

Run

```
  if (isEmpty(q)) {
    printf("Queue is empty");
    item = -1;
  } else {
    item = q->items[q->front];
    q->front++;
    if (q->front > q->rear) {
      printf("Resetting queue ");
      q->front = q->rear = -1;}}
  return item;}
void printQueue(struct queue* q) {
  int i = q->front;
  if (isEmpty(q)) {
    printf("Queue is empty");
  } else {
    printf("\nQueue contains \n");
    for (i = q->front; i < q->rear + 1; i++) {
      printf("%d ", q->items[i]);}}}
int main() {
  struct Graph* graph = createGraph(6);
  addEdge(graph, 0, 1);
  addEdge(graph, 0, 2);
  addEdge(graph, 1, 2);
  addEdge(graph, 1, 4);
  addEdge(graph, 1, 3);
  addEdge(graph, 2, 4);
  addEdge(graph, 3, 4);
  bfs(graph, 0);
  return 0;
}
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Queue contains
0 Resetting queue Visited 0

Queue contains
2 1 Visited 2

Queue contains
1 4 Visited 1

Queue contains
4 3 Visited 4

Queue contains
3 Resetting queue Visited 3
```