1.

	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	0
C	1	0	0	1	1
D	1	1	1	0	1
E	0	0	1	1	0

A -> B -> C ->D

B -> A -> D

C -> A -> D -> E

D -> A -> B -> C -> E

E -> C -> D

	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

0 -> 1 -> 2 ->3 -> 4

1 -> 0 -> 3

2 -> 0 -> 3 -> 4 -> 5

3 -> 0 -> 1 -> 2 -> 5

4 -> 0 -> 2

5 -> 2 -> 3

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	1
C	0	1	0	1	0
D	0	0	1	0	1
E	1	1	0	1	0

A -> B -> E

B -> A -> C -> E

C -> B -> D

D -> C -> E

E -> A -> B -> D

	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

A -> B -> C -> D

B -> A -> D -> E

C -> A -> D

D -> A -> B -> C -> D -> E

E -> B -> D

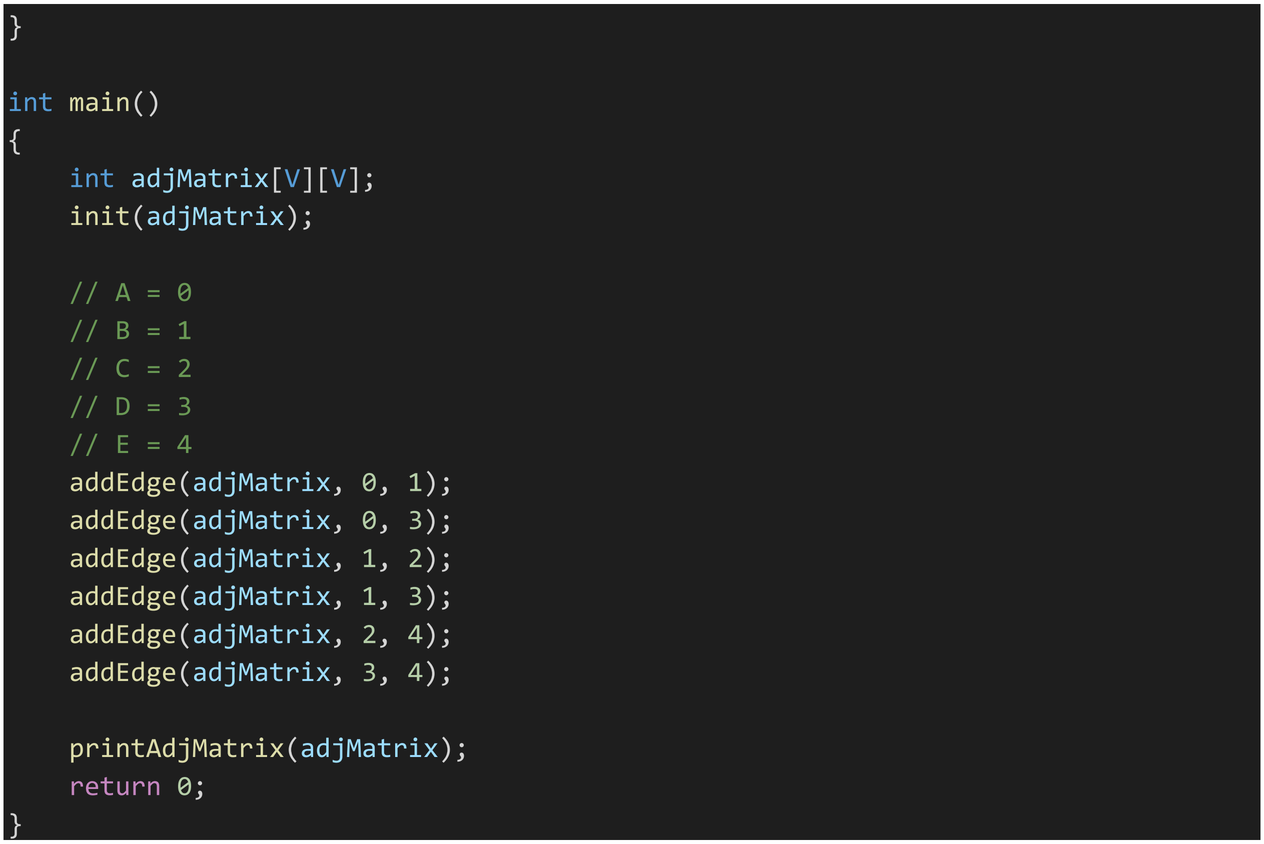
1.

a.

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

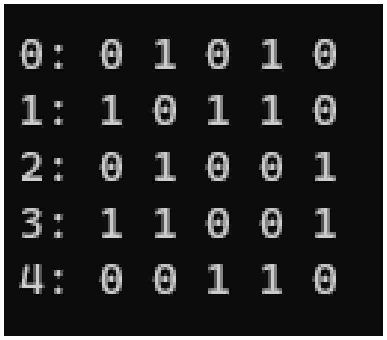
b.

```c
#include <stdio.h>
#define V 5

// initialize the matrix
void init(int arr[][V])
{
    int i, j;
    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            arr[i][j] = 0;
}

// add edges
void addEdge(int arr[][V], int i, int j)
{
    arr[i][j] = 1;
    arr[j][i] = 1;
}

// print the matrix
void printAdjMatrix(int arr[][V])
{
    int i, j;
    for (i = 0; i < V; i++)
    {
        printf("%d: ", i);
        for (j = 0; j < V; j++)
        {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
```



```
}

int main()
{
    int adjMatrix[V][V];
    init(adjMatrix);

    // A = 0
    // B = 1
    // C = 2
    // D = 3
    // E = 4
    addEdge(adjMatrix, 0, 1);
    addEdge(adjMatrix, 0, 3);
    addEdge(adjMatrix, 1, 2);
    addEdge(adjMatrix, 1, 3);
    addEdge(adjMatrix, 2, 4);
    addEdge(adjMatrix, 3, 4);

    printAdjMatrix(adjMatrix);
    return 0;
}
```

c.



```
0: 0 1 0 1 0
1: 1 0 1 1 0
2: 0 1 0 0 1
3: 1 1 0 0 1
4: 0 0 1 1 0
```

2.

a.

A -> B -> D

B -> A -> C -> D

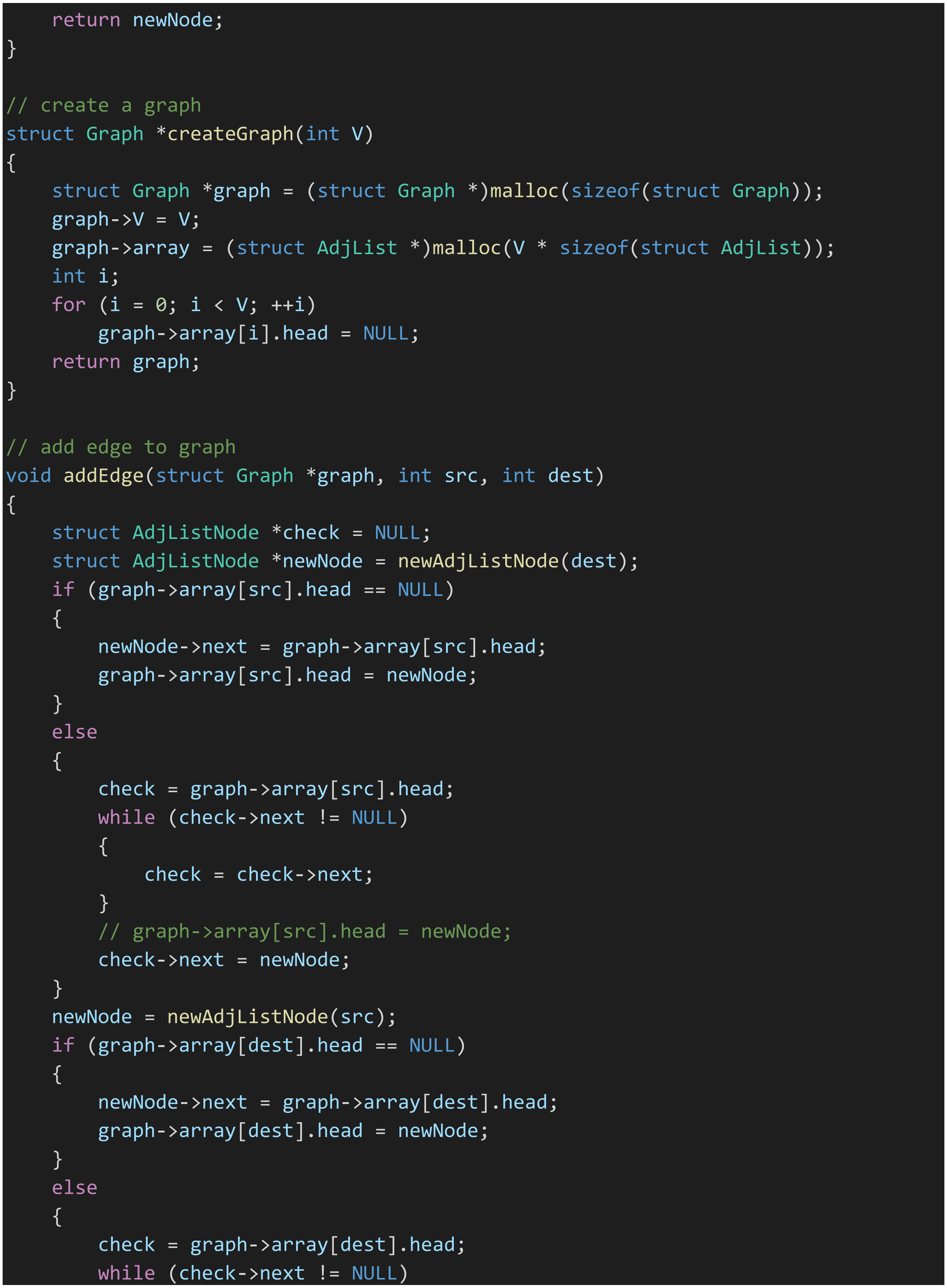
C -> B -> E

D -> A -> B -> E

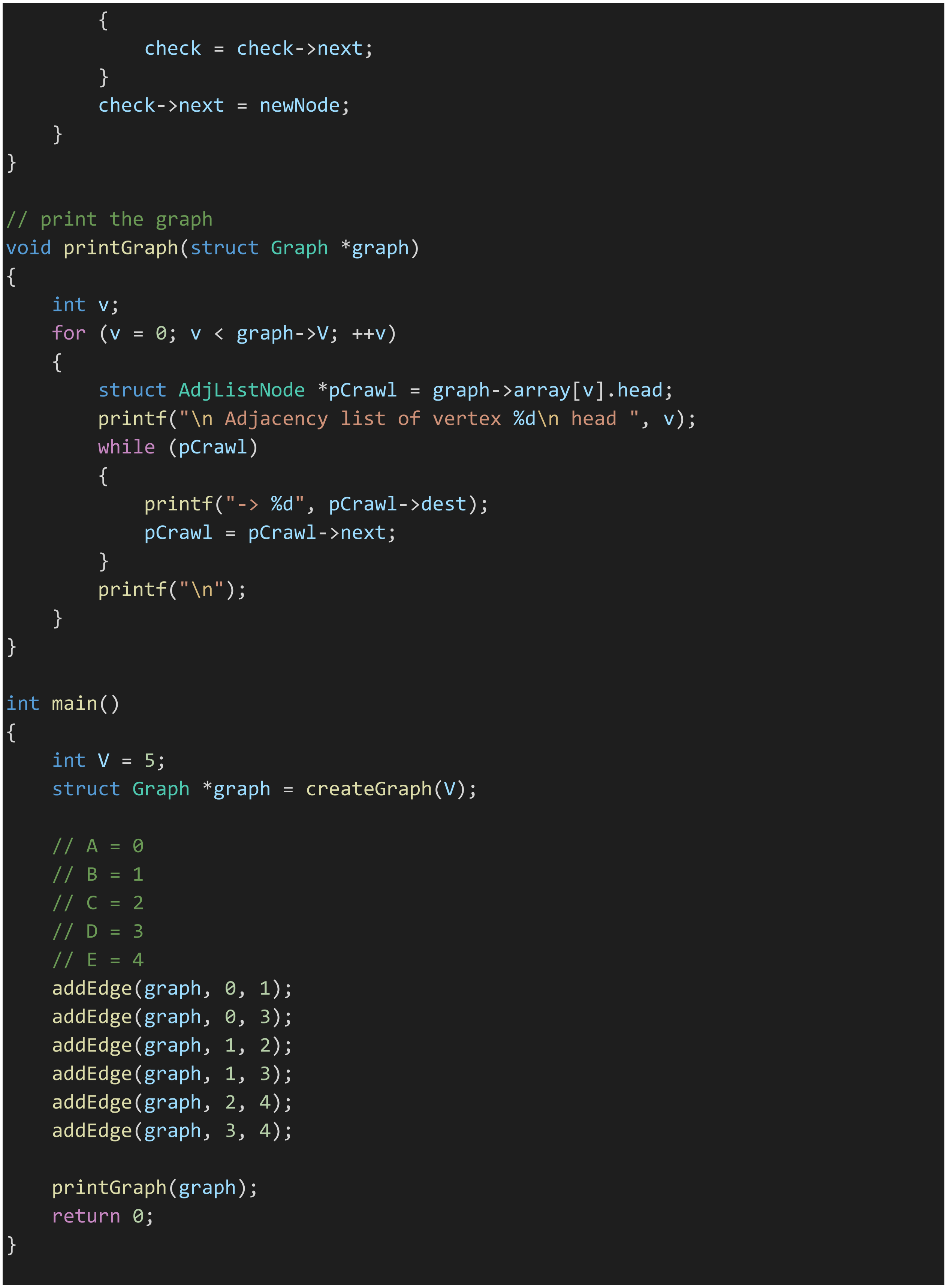
E -> C -> D

b.

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

// structure to adjacency list node
struct AdjListNode
{
    int dest;
    struct AdjListNode *next;
};

// structure to adjacency list
struct AdjList
{
    struct AdjListNode *head;
};

// structure to graph
struct Graph
{
    int V;
    struct AdjList *array;
};

// add list nodes
struct AdjListNode *newAdjListNode(int dest)
{
    struct AdjListNode *newNode = (struct AdjListNode *)malloc(sizeof(struct
AdjListNode));
    newNode->dest = dest;
    newNode->next = NULL;
```

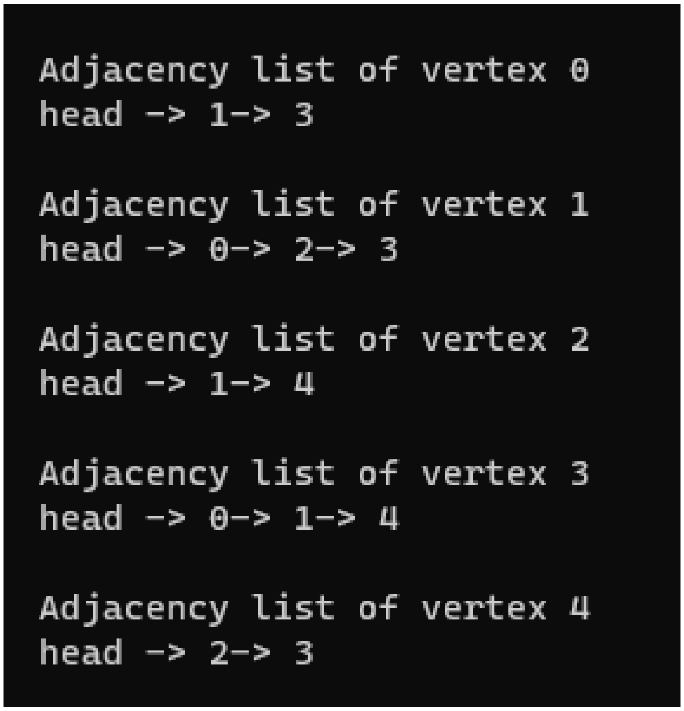


```
    return newNode;
}

// create a graph
struct Graph *createGraph(int V)
{
    struct Graph *graph = (struct Graph *)malloc(sizeof(struct Graph));
    graph->V = V;
    graph->array = (struct AdjList *)malloc(V * sizeof(struct AdjList));
    int i;
    for (i = 0; i < V; ++i)
        graph->array[i].head = NULL;
    return graph;
}

// add edge to graph
void addEdge(struct Graph *graph, int src, int dest)
{
    struct AdjListNode *check = NULL;
    struct AdjListNode *newNode = newAdjListNode(dest);
    if (graph->array[src].head == NULL)
    {
        newNode->next = graph->array[src].head;
        graph->array[src].head = newNode;
    }
    else
    {
        check = graph->array[src].head;
        while (check->next != NULL)
        {
            check = check->next;
        }
        // graph->array[src].head = newNode;
        check->next = newNode;
    }
    newNode = newAdjListNode(src);
    if (graph->array[dest].head == NULL)
    {
        newNode->next = graph->array[dest].head;
        graph->array[dest].head = newNode;
    }
    else
    {
        check = graph->array[dest].head;
        while (check->next != NULL)
```



```c
        {
            check = check->next;
        }
        check->next = newNode;
    }
}

// print the graph
void printGraph(struct Graph *graph)
{
    int v;
    for (v = 0; v < graph->V; ++v)
    {
        struct AdjListNode *pCrawl = graph->array[v].head;
        printf("\n Adjacency list of vertex %d\n head ", v);
        while (pCrawl)
        {
            printf("-> %d", pCrawl->dest);
            pCrawl = pCrawl->next;
        }
        printf("\n");
    }
}

int main()
{
    int V = 5;
    struct Graph *graph = createGraph(V);

    // A = 0
    // B = 1
    // C = 2
    // D = 3
    // E = 4
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 3);
    addEdge(graph, 1, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 2, 4);
    addEdge(graph, 3, 4);

    printGraph(graph);
    return 0;
}
```

c.



```
Adjacency list of vertex 0
head -> 1-> 3

Adjacency list of vertex 1
head -> 0-> 2-> 3

Adjacency list of vertex 2
head -> 1-> 4

Adjacency list of vertex 3
head -> 0-> 1-> 4

Adjacency list of vertex 4
head -> 2-> 3
```