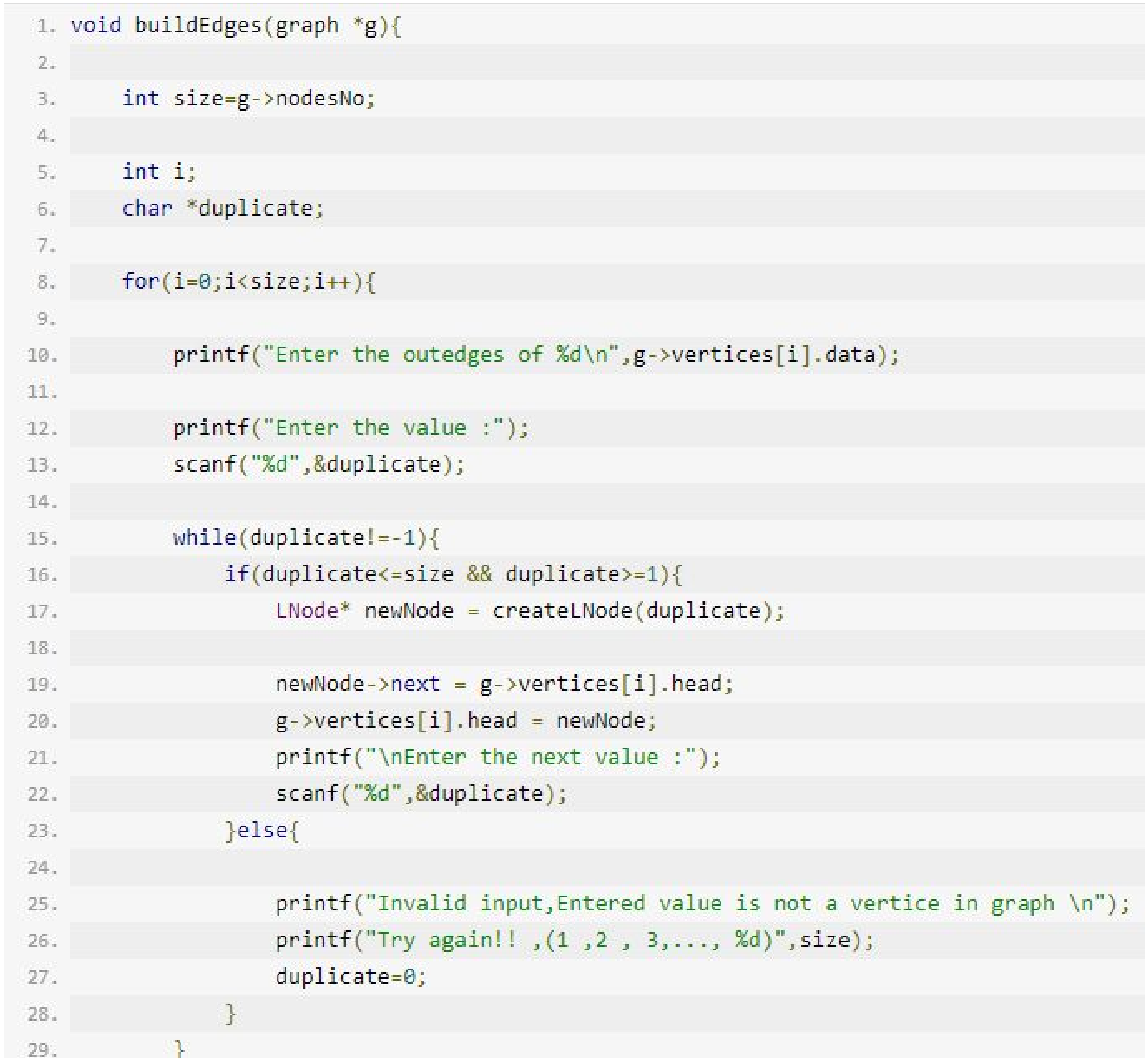
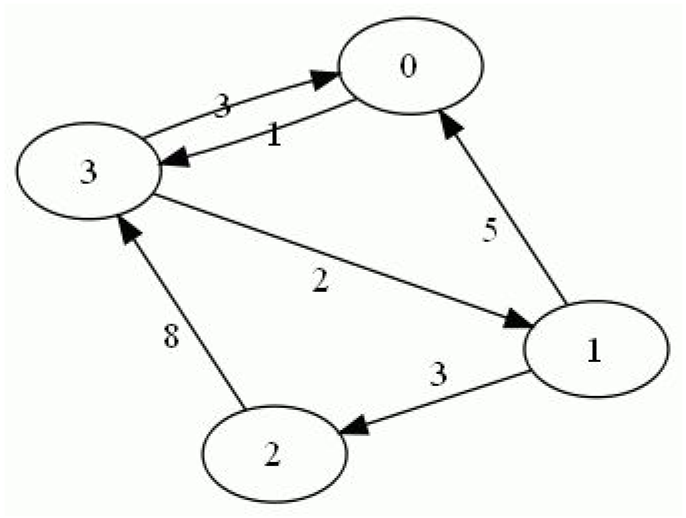
Directed Graph Implementation

This is regarding “MyGraphImplementation.c” .Its a code which implement a directed graph according to the given user number of inputs.In this it’s automatically assign numbers as its data upto given number of inputs.

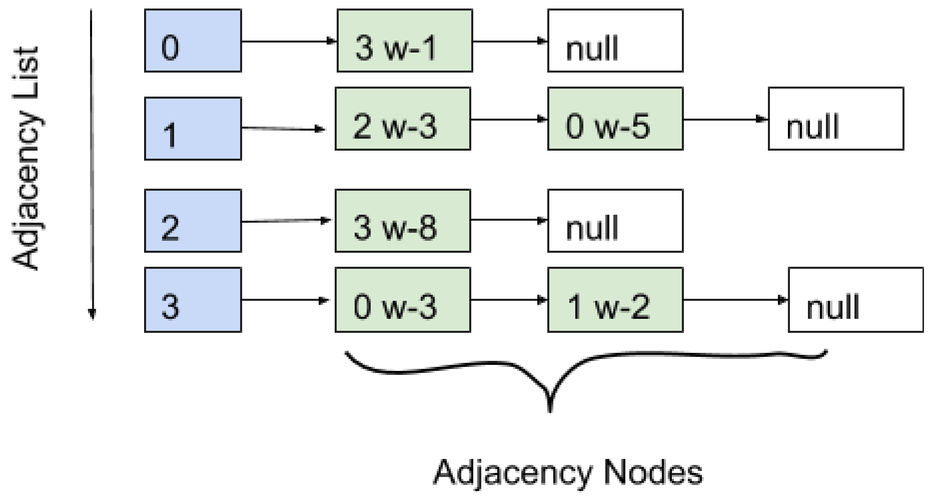


```
void buildEdges(graph *g){

    int size=g->nodesNo;

    int i;
    char *duplicate;

    for(i=0;i<size;i++){

        printf("Enter the outedges of %d\n",g->vertices[i].data);

        printf("Enter the value :");
        scanf("%d",&duplicate);

        while(duplicate!=-1){
            if(duplicate<=size && duplicate>=1){
                LNode* newNode = createLNode(duplicate);

                newNode->next = g->vertices[i].head;
                g->vertices[i].head = newNode;
                printf("\nEnter the next value :");
                scanf("%d",&duplicate);
            }else{

                printf("Invalid input,Entered value is not a vertice in graph \n");
                printf("Try again!! ,(1 ,2 , 3,..., %d)",size);
                duplicate=0;
            }
        }
```

To a better explanation consider following example.



In this weighted directed graph adjacency list representation looks like below.



In here adjacency nodes hold the weight and adjacency list has a pointer to nodes.

```
struct AdListNode{
    int val;
    int weight;
    struct AdListNode *next;
};typedef struct AdListNode LNode;
```

This is code to build adjacency node and there is pointer to next node as well

```
struct AdList{
    int data;
    LNode *head;
};typedef struct AdList adlist;
```

In this the data variable holds value we enter as the vertice
According to the above example they are 0,1,2,3

```c
struct GRAPH{
    int nodesNo;
    adlist *vertices;
};typedef struct GRAPH graph;
```

nodesNo is the variable which holds value of total number of nodes in graph. Vertices is the name given by me to the adjacency list.

```c
graph * createGraph(void){

    int input;
    printf("Enter the num of nodes you need : ");
    scanf("%d",&input);

    graph *g=(graph*)malloc(sizeof(graph));
    g->nodesNo=input;

    g->vertices=(adlist*)malloc(input * (sizeof(adlist)));

    int i;

    //this will automatically create vertices as 1,2,3,4,.....,input
    for(i=1;i<=input;i++){
        g->vertices[i-1].data=i;
        g->vertices[i-1].head=NULL;
    }

    //this will print verices in the graph
    /*for(i=1;i<=input;i++){
        printf("check graph: %d \n",(g->vertices[i-1].data));
    }*/
    return g;
}
```

This creates graph vertices.Actually it creates Adjacency list to hold its adjacency nodes which connected with it.
Here when number nodes entered the ***for loop*** creates vertices with values 1 to ***"input"***

This function is to create adjacency nodes.Here it takes a ***graph pointer*** as an input and allow user to enter values that can connected to given vertice.

Here you can use only the values among ***1 to number of inputs*** in the graph which you created before.

Prerequisites for Code to work:

- In here to implement the graph you should know the exact number of vertices you have.
- The one who execute the program cannot give ***"-1"*** as an adjacency node in this graph implementation because it's the terminating value of ***while loop.***
- ***"0"***cannot be included in this graph because ***createGraph*** **function** always **generates values from starting 1**