Directed Graph Input Without Any Weight

```
#include <bits/stdc++.h>
// a header file that includes every standard library

using namespace std;

// Vector Reference
// https://www.cplusplus.com/reference/vector/vector/
vector<int>graph_test [10];


int main() {

    // random variables
    int i, j;

    // graph input variables
    int num_of_edge = 3;
    int num_of_node = 4;
    int start_point, end_point;

    for(i=1; i<=num_of_edge; i++)
    {
        cout<<"Edge: "<<i<<endl;
        cin>>start_point;
        cin>>end_point;

        //for the connection (edge) between two nodes
        graph_test[start_point].push_back(end_point);
    }

    //testing purpose only

    cout<<graph_test[1].size()<<endl;

    for(i=1; i<=num_of_node; i++)
    {
        cout<<"node--"<<i<<" "<<"Connected to"<<endl;
        for(j=0; j<graph_test[i].size(); j++)
        {
            cout<<graph_test[i][j]<<endl;
        }
    }

    return 0;
}
```

Directed Graph Input With Weight

```
#include <bits/stdc++.h>


using namespace std;


vector<pair<int,int>> graph_test_pair[10];

// or

/*
typedef pair<int,int> c_pair;
vector<c_pair> graph_test_pair[10];
*/

int main() {

    // random variables
    int i, j;

    // graph input variables
    int num_of_edge = 3;
    int num_of_node = 4;
    int start_point, end_point, weight;

    // graph input with weight
    for(i=1; i<=num_of_edge; i++)
    {
        cout<<"Edge: "<<i<<endl;
        cin>>start_point;
        cin>>end_point;
        cin>>weight;

        //for the connection (edge) between two nodes with weight
        graph_test_pair[start_point].push_back(make_pair(end_point, weight));

    }


    //testing purpose only
    cout<<"test output"<<endl;
    cout<<graph_test_pair[1].size()<<endl;
    cout<<"Sample Edge: "<<graph_test_pair[1][1].first<<endl;
    cout<<"Sample Weight"<<graph_test_pair[1][1].second<<endl;


    // print your graph with weights
    for(i=1; i<=num_of_node; i++)
    {
```

```
        cout<<"node--"<<i<<" "<<"Connected to"<<endl;
        for(j=0; j<graph_test_pair[i].size(); j++)
        {
            cout<<graph_test_pair[i][j].first<<endl;
            cout<<graph_test_pair[i][j].second<<endl;
        }
    }

    return 0;
}
```

Deque demo by Eduardo Corpeño

a double-ended queue (abbreviated to **deque**) for which elements can be added to or removed from either the front (head) or back (tail).

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    deque<int> numbers;

    int temp=0;

    cout<<"Pushing Back..."<<endl;
    while(temp>=0){
        cout<<"Enter Number: ";
        cin>>temp;
        if(temp>=0)
            numbers.push_back(temp);
    }

    deque<int>::iterator it;
    cout<<"{  ";
    for(it = numbers.begin(); it!=numbers.end(); it++)
        cout<<*it<<"  ";
    cout<<"}";

    temp=0;
    cout<<endl;
    cout<<"Pushing Front..."<<endl;
    while(temp>=0){
        cout<<"Enter Number: ";
        cin>>temp;
        if(temp>=0)
            numbers.push_front(temp);
    }

    cout<<"{  ";
```

```
    for(it = numbers.begin(); it!=numbers.end(); it++)
        cout<<*it<<"  ";
    cout<<"}";


    temp=0;
    cout<<endl;
    cout<<"Pushing Back..."<<endl;
    while(temp>=0){
        cout<<"Enter Number: ";
        cin>>temp;
        if(temp>=0)
            numbers.push_back(temp);
    }

    cout<<"Current Status: "<<endl;
    cout<<"{  ";
    for(it = numbers.begin(); it!=numbers.end(); it++)
        cout<<*it<<"  ";
    cout<<"}"<<endl;

    cout<<"Pop Back..."<<endl;
    numbers.pop_back();

    cout<<"{  ";
    for(it = numbers.begin(); it!=numbers.end(); it++)
        cout<<*it<<"  ";
    cout<<"}"<<endl;

    cout<<"Pop Front..."<<endl;
    numbers.pop_front();

    cout<<"{  ";
    for(it = numbers.begin(); it!=numbers.end(); it++)
        cout<<*it<<"  ";
    cout<<"}"<<endl;


    return 0;
}
```

Stack using STL

[Reference Link]

```
#include<bits/stdc++.h>
using namespace std;

int main(){
```

```
    stack<int> numbers;
    int temp;

    cout<<"Pushing..."<<endl;
    while(temp>=0){
        cout<<"Enter numbers: ";
        cin>>temp;
        if(temp>=0)
            numbers.push(temp);
    }

    cout<<"{  ";
    while(numbers.size()>0){
        cout<<numbers.top();
        numbers.pop();
        cout<<"  ";
    }
    cout<<"}";
    return 0;
}
```

Queue

Reference Link

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    queue<int> numbers;
    int temp;
    cout<<"Pushing..."<<endl;
    while(temp>=0){
        cout<<"Enter numbers: ";
        cin>>temp;
        if(temp>=0)
            numbers.push(temp);
    }
    cout<<"{  ";
    while(numbers.size()>0){
        cout<<numbers.front();
        numbers.pop();
        cout<<"  ";
    }
    cout<<"}";
    return 0;
}
```

Priority Queue

[Reference Link]

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  priority_queue<int> numbers;
  int temp;

  cout<<"Pushing...\n";
  while(temp>=0){
    cout<<"Enter numbers: ";
    cin>>temp;
    if(temp>=0)
      numbers.push(temp);
  }

  cout<<"{  ";
  while(numbers.size()>0){
    cout<<numbers.top();
    numbers.pop();
    cout<<"  ";
  }
  cout<<"}";
  return 0;
}
```

Back Button by Eduardo Corpeño stack usage example

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  stack<string> back_stack;
  string temp;
  while(temp!="exit"){
    cout<<"[1] Visit URL   [2] Back"<<endl;
    cin>>temp;
    if(temp=="exit")
      break;
    if(temp=="1"){
      cout<<"Enter URL: ";
      cin>>temp;
      back_stack.push(temp);
    }
    else if(temp=="2"){
      cout<<"Going back...\n";
```