**Finding the shortest path between two nodes:**

// BFS searches level-wise, so it is useful here

// Assuming the adj list, arrays are declared globally (if not, pass them also in the function)

bool BFS\_shortest\_path (int source, int destination) {

mark source as visited

insert the source in a queue

while (queue is not empty) {

curr= front element of queue

pop it out

for (traverse the children) {

if (child not visited) {

mark visited

push it into the queue

in the predecessor array, insert the ‘curr’ node at this child’s index

if the child is the destination, return true (indicating a path exists)

}

}

}

// if it doesn’t return above, return false here (indicating the path doesn’t exist)

}

// This is what to do in the main function

int main () {

// Take inputs

// A vector for storing the path

int curr=destination;

push curr into the path (it will be stored in reverse order, so print from behind)

while (predecessor of curr! = -1) {

// the predecessor of source was initialized to -1

push back predecessor of curr;

curr becomes predecessor of current

}

}

// print the vector in reverse order to get the path