/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* An example of creating an array of lists dynamically.

\* This code can be modified to create adjacency list representation of a graph.

\*

\* Written for the ESE344 by Prof. Murali Subbarao

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <assert.h>

using namespace std;

struct node {

int vertex; // public data members

node \*next;

node(int k, node \*p) { // constructor

vertex = k;

next = p;

}

};

int main()

{

int size1=10, size2=8; // these can be read as input at run time.

// create an array of pointers to the first node of lists

node \*\*lst;

lst = new node\*[size1]; // an array of pointers of size1

for (int i = 0; i < size1; i++) { // initialize values to null

lst[i] = nullptr;

}

// creating and initializing an array of lists

int count = 0;

for (int i = 0; i < size1; i++) { // for each list i

for (int j = 0; j < size2; j++) // for each node j in list i

{

node \*p;

p = new node(count++, lst[i]); // get a new node with required initial values

lst[i] = p; // add the new node at the beginning of the list i

}

}

// print the list created above

for (int i = 0; i < size1; i++) {

for (node \*p = lst[i]; p != nullptr; p = p->next)

{

cout << p->vertex << " ";

}

cout << endl;

}

// erase the list

for (int i = 0; i < size1; i++) { // delete this list

for (node \*p = lst[i]; p != nullptr;)

{

node \*tmp = p;

p = p->next;

delete tmp; // delete this node

}

}

delete lst; // delete array of pointers

cout << "Enter any number to exit : ";

cin >> size1; // make the program wait till this input is entered (otherwise the output window disappears)

return 0;

}