#include

<stdio.h>

Using namespace std;

#define N 4

Struct Node

{

Node\* parent;

Int pathCost;

Int cost;

Int workerID;

Int jobID;

Bool assigned[N];

};

Node\* newNode(int x, int y, bool assigned[],

Node\* parent)

{

Node\* node = new Node;

For (int j = 0; j < N; j++)

Node->assigned[j] = assigned[j];

Node->assigned[y] = true;

node->parent = parent;

node->workerID = x;

node->jobID = y;

return node;

}

int calculateCost(int costMatrix[N][N], int x,

int y, bool assigned[])

{

int cost = 0;

bool available[N] = {true};

for (int i = x + 1; i < N; i++)

{

int min = INT\_MAX, minIndex = -1;

for (int j = 0; j < N; j++)

{

if (!assigned[j] && available[j] &&

costMatrix[i][j] < min)

{

minIndex = j;

min = costMatrix[i][j];

}

}

cost += min;

available[minIndex] = false;

}

return cost;

}

struct comp

{

bool operator()(const Node\* lhs,

const Node\* rhs) const

{

return lhs->cost > rhs->cost;

}

};

void printAssignments(Node \*min)

{

if(min->parent==NULL)

return;

printAssignments(min->parent);

cout << "Assign Worker " << char(min->workerID + 'A')

<< " to Job " << min->jobID << endl;

}

int findMinCost(int costMatrix[N][N])

{

priority\_queue<Node\*, std::vector<Node\*>, comp> pq;

bool assigned[N] = {false};

Node\* root = newNode(-1, -1, assigned, NULL);

root->pathCost = root->cost = 0;

root->workerID = -1;

pq.push(root);

while (!pq.empty())

{

Node\* min = pq.top();

pq.pop();

int i = min->workerID + 1;

if (i == N)

{

printAssignments(min);

return min->cost;

}

for (int j = 0; j < N; j++)

{

if (!min->assigned[j])

{

Node\* child = newNode(i, j, min->assigned, min);

child->pathCost = min->pathCost + costMatrix[i][j];

child->cost = child->pathCost +

calculateCost(costMatrix, i, j, child->assigned);

pq.push(child);

}

}

}

}