**Problem statement:**

**Given N users, and for each user we have a set of intervals to which this user can be assigned. Assign intervals to the users such that:**

**An interval is assigned to only one user.**

**All users are assigned an interval. If it is not possible to assign to all users, then assign to maximum number of users.**

**Ex-**

**A → I1, I2, I3**

**B → I1**

**C → I2, I3**

**D → I2**

**One probable answer:**

**A → I2**

**B → I1**

**C → I3**

**D→ cannot be assigned**

**Note:**

**Interval (I) contains:**

**intervalId : integer**

**startTime : date-time**

**endTime : date-time**

**User (U) contains:**

**userId: integer**

**username : string**

**1) Using Backtracking Algorithm**

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

**using namespace std;**

**const int N=1e2+5;**

**vector<vector<int>> FeasibleInterVals(N);**

**vector<int> UserAssigned(N,-1);**

**vector<int> InterValAssigned(N,-1);**

**vector<vector<pair<int,int>>> Solution;**

**void assignInterVals(int userId, int noUser)**

**{ //termination condition arrived and store the solution**

**if(userId==noUser+1)**

**{**

**vector<pair<int,int>> PossibleSolution;**

**for(int user=1;user<=noUser;user++)**

**{**

**if(UserAssigned[user]>0)**

**PossibleSolution.push\_back(make\_pair(user,UserAssigned[user]));**

**}**

**Solution.push\_back(PossibleSolution);**

**//deleting smaller solutions**

**for(auto oneSolution =Solution.begin();oneSolution!=Solution.end();oneSolution++)**

**{**

**if(oneSolution->size()<PossibleSolution.size())**

**{**

**Solution.erase(oneSolution);**

**oneSolution--;**

**}**

**}**

**return;**

**}**

**//excluding any assignment**

**assignInterVals(userId+1,noUser);**

**for(int interval : FeasibleInterVals[userId])**

**{ //already assigend hence continue**

**if(InterValAssigned[interval]!=-1) continue;**

**UserAssigned[userId]=interval;**

**InterValAssigned[interval]=userId;**

**assignInterVals(userId+1,noUser);**

**//backtracking**

**UserAssigned[userId]=-1;**

**InterValAssigned[interval]=-1;**

**}**

**}**

**bool operator<(vector<pair<int,int> >& p, vector<pair<int,int> > & q)**

**{**

**return (p.size()<q.size());**

**}**

**void PrintSolution()**

**{ //removing duplicate Solutions**

**auto iterAtor=unique(Solution.begin(),Solution.end());**

**Solution.resize(iterAtor-Solution.begin());**

**//Printing the Optimal Solutions**

**for(int ithSolution=0;ithSolution <min(4,(int)Solution.size());ithSolution++)**

**{**

**vector<pair<int,int>> oneSolution =Solution[ithSolution];**

**sort(oneSolution.begin(),oneSolution.begin());**

**for(int user=0;user<oneSolution.size();user++)**

**{**

**cout<<oneSolution[user].first<<"th user is assigned "<<oneSolution[user].second<<"th interval"<<endl;**

**}**

**cout<<endl;**

**}**

**}**

**int main()**

**{**

**time\_t start, end;**

**time(&start);**

**int noUser;**

**//enter the number of users**

**cin >> noUser;**

**int noInterval;**

**//enter the number of intervals**

**cin >> noInterval;**

**for(int user=1;user<=noUser;user++)**

**{**

**//enter the number of favaroble intervals of ith user**

**int noFeasibleIntervals; cin >> noFeasibleIntervals;**

**while(noFeasibleIntervals--)**

**{**

**//enter the interval for ith user**

**int ithFavourableInterval;**

**cin >> ithFavourableInterval;**

**FeasibleInterVals[user].push\_back(ithFavourableInterval);**

**}**

**}**

**assignInterVals(1,noUser);**

**PrintSolution();**

**time(&end);**

**double time\_taken = double(end - start);**

**cout << "Time taken by program is : " << fixed << time\_taken << setprecision(5);**

**cout << " sec " << endl;**

**return 0;**

**}**

**2) Using Ford Fulkerson Algorithm**

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

**using namespace std;**

**const int N=1e2+5;**

**vector<int> parent(N);**

**vector<int> visited(N);**

**vector<vector<pair<int,int> > > adjacencyList(N);**

**vector<vector<pair<int,int> > > residualAdjacencyList(N);**

**vector<vector<int> > input(N);**

**vector<vector<pair<int,int> > > solution;**

**int totalAssignment=10000;**

**int finalEdge=10000;**

**bool comparTor(vector<vector<pair<int,int> >> ::iterator iteraTor1, vector<vector<pair<int,int>> > ::iterator iteraTor2)**

**{**

**vector<pair<int,int> > vector1=\*iteraTor1;**

**vector<pair<int,int> >vector2=\*iteraTor2;**

**if(vector1.size()!=vector2.size())**

**return false;**

**for(int onesolutionVariable =0; onesolutionVariable<vector1.size(); onesolutionVariable++)**

**{**

**if(vector1[onesolutionVariable].first==vector2[onesolutionVariable].first and vector1[onesolutionVariable].second == vector2[onesolutionVariable].second)**

**continue;**

**return false;**

**}**

**return true;**

**}**

**void removeDuplicates()**

**{**

**for(vector<vector<pair<int,int >>> :: iterator iteraTor =solution.begin();iteraTor != solution.end(); iteraTor++)**

**{**

**for(vector<vector<pair<int,int> >> :: iterator itr=solution.begin();itr!=iteraTor;itr++ )**

**{**

**if(comparTor(iteraTor,itr))**

**{**

**solution.erase(iteraTor);**

**iteraTor--;**

**break;**

**}**

**}**

**}**

**}**

**void reverseSort()**

**{**

**for (int iteraingVariable = solution.size() - 2; iteraingVariable >= 0; --iteraingVariable)**

**{**

**vector<pair<int,int> > oneSolution=solution[iteraingVariable];**

**int iteratingVariable2=iteraingVariable;**

**while (iteratingVariable2 < solution.size() - 1 && oneSolution.size() > solution[iteratingVariable2].size())**

**{**

**solution[iteratingVariable2]=solution[iteratingVariable2+1];**

**iteratingVariable2++;**

**}**

**solution[iteratingVariable2]=oneSolution;**

**}**

**}**

**void printSolution()**

**{**

**reverseSort();**

**for(int iteratingVariable=0;iteratingVariable<min(100,(int)solution.size());iteratingVariable++)**

**{**

**vector<pair<int,int > > oneSolution = solution[iteratingVariable];**

**for(pair<int,int> onePair : oneSolution)**

**{**

**cout<<onePair.first<<"th user is assigned "<<onePair.second<<"th interval"<<endl;**

**}**

**cout<<endl;**

**}**

**}**

**int bfsForFindingParent(int sourceVertex, int destinationVertex)**

**{ //clearing the visited array**

**for(int node=0;node<N;node++)**

**visited[node]=0;**

**visited[sourceVertex]=1;**

**parent[sourceVertex]=-1;**

**queue<int> pendingNodes;**

**pendingNodes.push(sourceVertex);**

**while(!pendingNodes.empty())**

**{**

**int currentVertex=pendingNodes.front();**

**pendingNodes.pop();**

**for(pair<int,int> adjacentEdge : residualAdjacencyList[currentVertex])**

**{**

**int adjacentVertex=adjacentEdge.first;**

**int weightOfEdge=adjacentEdge.second;**

**if(visited[adjacentVertex]==0 and weightOfEdge>0)**

**{**

**pendingNodes.push(adjacentVertex);**

**visited[adjacentVertex]=1;**

**parent[adjacentVertex]=currentVertex;**

**}**

**}**

**}**

**return visited[destinationVertex];**

**}**

**void findingMaximumMapping(int noUser, int noInterval)**

**{**

**totalAssignment=0;**

**while( bfsForFindingParent(0,noUser+noInterval+2))**

**{**

**int possibleAssignment=INT\_MAX;**

**for(int node=noUser+noInterval+2;parent[node]!=-1;node=parent[node])**

**{**

**for(auto edge : residualAdjacencyList[parent[node]])**

**{**

**if(edge.first==node)**

**{**

**possibleAssignment=min(possibleAssignment,edge.second);**

**}**

**}**

**}**

**for(int node=noUser+noInterval+2;parent[node]!=-1 ;node=parent[node])**

**{**

**for(pair<int,int>& edge : residualAdjacencyList[parent[node]])**

**{**

**if(edge.first==node)**

**{**

**edge.second-=possibleAssignment;**

**}**

**}**

**for(pair<int,int>& edge : residualAdjacencyList[node])**

**{**

**if(edge.first==parent[node])**

**edge.second+=possibleAssignment;**

**}**

**}**

**totalAssignment+=possibleAssignment;**

**}**

**}**

**void initialisingAdjacencyList(int noUser ,int noInterval)**

**{**

**for(int node=0;node<=noUser+noInterval+2;node++)**

**{**

**residualAdjacencyList[node].clear();**

**adjacencyList[node].clear();**

**}**

**for(int user=1;user<=noUser;user++)**

**{**

**adjacencyList[0].push\_back(make\_pair(user,1));**

**residualAdjacencyList[0].push\_back(make\_pair(user,1));**

**//reverse edge with zero weight**

**adjacencyList[user].push\_back(make\_pair(0,0));**

**residualAdjacencyList[user].push\_back(make\_pair(0,0));**

**}**

**for(int interval =noUser+1;interval<=noUser+noInterval;interval++)**

**{**

**adjacencyList[interval].push\_back(make\_pair(noUser+noInterval+1,1));**

**residualAdjacencyList[interval].push\_back(make\_pair(noUser+noInterval+1,1));**

**adjacencyList[noUser+noInterval+1].push\_back(make\_pair(interval,0));**

**residualAdjacencyList[noUser+noInterval+1].push\_back(make\_pair(interval,0));**

**}**

**residualAdjacencyList[noUser+noInterval+1].push\_back(make\_pair(noUser+noInterval+2,finalEdge));**

**residualAdjacencyList[noUser+noInterval+2].push\_back(make\_pair(noUser+noInterval+1,0));**

**adjacencyList[noUser+noInterval+1].push\_back(make\_pair(noUser+noInterval+2,finalEdge));**

**adjacencyList[noUser+noInterval+2].push\_back(make\_pair(noUser+noInterval+1,0));**

**}**

**void userIntervalMapping(int noUser, int noInterval)**

**{**

**initialisingAdjacencyList(noUser,noInterval);**

**for(int user=1;user<=noUser;user++)**

**{**

**for(int interval : input[user])**

**{**

**adjacencyList[user].push\_back(make\_pair(interval+noUser,1));**

**residualAdjacencyList[user].push\_back(make\_pair(interval+noUser,1));**

**// //adding reverse edge**

**adjacencyList[interval+noUser].push\_back(make\_pair(user,0));**

**residualAdjacencyList[interval+noUser].push\_back(make\_pair(user,0));**

**}**

**}**

**findingMaximumMapping(noUser,noInterval);**

**vector<pair<int,int> > oneSolution;**

**for(int user=1;user<=noUser;user++)**

**{**

**int \* freqWeight= new int[105]{};**

**for(pair<int,int> edge : adjacencyList[user]) freqWeight[edge.first]+=edge.second;**

**for(pair<int,int> edge : residualAdjacencyList[user]) freqWeight[edge.first]-=edge.second;**

**for(int frequency =noUser+1;frequency<=noUser+noInterval;frequency++)**

**{**

**if(freqWeight[frequency]==1)**

**{**

**oneSolution.push\_back(make\_pair(user,frequency-noUser));**

**}**

**}**

**}**

**sort(oneSolution.begin(),oneSolution.end());**

**solution.push\_back(oneSolution);**

**removeDuplicates();**

**}**

**void userReverseIntervalMapping(int noUser, int noInterval)**

**{**

**initialisingAdjacencyList(noUser,noInterval);**

**for(int user=1;user<=noUser;user++)**

**{**

**for(int interval : input[user])**

**{**

**adjacencyList[noUser+1-user].push\_back(make\_pair(interval+noUser,1));**

**residualAdjacencyList[noUser+1-user].push\_back(make\_pair(interval+noUser,1));**

**// //adding reverse edge**

**adjacencyList[interval+noUser].push\_back(make\_pair(noUser-user+1,0));**

**residualAdjacencyList[interval+noUser].push\_back(make\_pair(noUser-user+1,0));**

**}**

**}**

**findingMaximumMapping(noUser,noInterval);**

**vector<pair<int,int> > oneSolution;**

**for(int user=1;user<=noUser;user++)**

**{**

**int \* freqWeight= new int[105]{};**

**for(pair<int,int> edge : adjacencyList[user]) freqWeight[edge.first]+=edge.second;**

**for(pair<int,int> edge : residualAdjacencyList[user]) freqWeight[edge.first]-=edge.second;**

**for(int frequency =noUser+1;frequency<=noUser+noInterval;frequency++)**

**{**

**if(freqWeight[frequency]==1)**

**{**

**oneSolution.push\_back(make\_pair(noUser-user+1,frequency-noUser));**

**}**

**}**

**}**

**sort(oneSolution.begin(),oneSolution.end());**

**solution.push\_back(oneSolution);**

**removeDuplicates();**

**}**

**void userIntervalReverseMapping(int noUser, int noInterval)**

**{**

**initialisingAdjacencyList(noUser,noInterval);**

**for(int user=1;user<=noUser;user++)**

**{**

**for(int interval : input[user])**

**{**

**adjacencyList[user].push\_back(make\_pair(noInterval-interval+noUser+1,1));**

**residualAdjacencyList[user].push\_back(make\_pair(noInterval-interval+noUser+1,1));**

**// //adding reverse edge**

**adjacencyList[noInterval-interval+noUser+1].push\_back(make\_pair(user,0));**

**residualAdjacencyList[noInterval-interval+noUser+1].push\_back(make\_pair(user,0));**

**}**

**}**

**findingMaximumMapping(noUser,noInterval);**

**vector<pair<int,int> > oneSolution;**

**for(int user=1;user<=noUser;user++)**

**{**

**int \* freqWeight= new int[105]{};**

**for(pair<int,int> edge : adjacencyList[user]) freqWeight[edge.first]+=edge.second;**

**for(pair<int,int> edge : residualAdjacencyList[user]) freqWeight[edge.first]-=edge.second;**

**for(int frequency =noUser+1;frequency<=noUser+noInterval;frequency++)**

**{**

**if(freqWeight[frequency]==1)**

**{**

**oneSolution.push\_back(make\_pair(user,noInterval+1-(frequency-noUser)));**

**}**

**}**

**}**

**sort(oneSolution.begin(),oneSolution.end());**

**solution.push\_back(oneSolution);**

**removeDuplicates();**

**}**

**void userReverseIntervalReverseMapping(int noUser, int noInterval)**

**{**

**initialisingAdjacencyList(noUser,noInterval);**

**for(int user=1;user<=noUser;user++)**

**{**

**for(int interval : input[user])**

**{**

**adjacencyList[noUser-user+1].push\_back(make\_pair(noInterval+1-interval+noUser,1));**

**residualAdjacencyList[noUser-user+1].push\_back(make\_pair(noInterval+1-interval+noUser,1));**

**// //adding reverse edge**

**adjacencyList[noInterval+1-interval+noUser].push\_back(make\_pair(noUser-user+1,0));**

**residualAdjacencyList[noInterval-interval+noUser+1].push\_back(make\_pair(noUser-user+1,0));**

**}**

**}**

**findingMaximumMapping(noUser,noInterval);**

**vector<pair<int,int> > oneSolution;**

**for(int user=1;user<=noUser;user++)**

**{**

**int \* freqWeight= new int[105]{};**

**for(pair<int,int> edge : adjacencyList[user]) freqWeight[edge.first]+=edge.second;**

**for(pair<int,int> edge : residualAdjacencyList[user]) freqWeight[edge.first]-=edge.second;**

**for(int frequency =noUser+1;frequency<=noUser+noInterval;frequency++)**

**{**

**if(freqWeight[frequency]==1)**

**{**

**oneSolution.push\_back(make\_pair(noUser+1-user,noInterval+1-(frequency-noUser)));**

**}**

**}**

**}**

**sort(oneSolution.begin(),oneSolution.end());**

**solution.push\_back(oneSolution);**

**removeDuplicates();**

**}**

**void userToIntervalMapping(int noUser ,int noInterval)**

**{ userIntervalMapping(noUser,noInterval);**

**userReverseIntervalMapping(noUser,noInterval);**

**userIntervalReverseMapping(noUser,noInterval);**

**userReverseIntervalReverseMapping(noUser,noInterval);**

**finalEdge=totalAssignment-1;**

**}**

**int main()**

**{**

**time\_t start, end;**

**time(&start);**

**// enter the number of users**

**int noUser;**

**cin >> noUser;**

**//enter the number of intervals**

**int noInterval;**

**cin >> noInterval;**

**//taking input**

**for(int user=1;user<=noUser;user++)**

**{ //enter number of intervals for ith user**

**int noOfIntervals;**

**cin >> noOfIntervals;**

**while(noOfIntervals--)**

**{**

**int favourableInterval;**

**cin >> favourableInterval;**

**input[user].push\_back(favourableInterval);**

**}**

**}**

**while(solution.size()<4 and totalAssignment>0)**

**userToIntervalMapping(noUser,noInterval);**

**printSolution();**

**time(&end);**

**double time\_taken = double(end - start);**

**cout<<"calculation of total time taken by program " << fixed << time\_taken << setprecision(5);**

**cout << " sec " << endl;**

**return 0;**

**}**

**Test Case 1 :-**

**A → I1, I2**

**B → I1, I2**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**A → I2**

**B → I1**

**total time taken by program 0.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**A → I2**

**B → I1**

**A → I1**

**B → I1**

**total time taken by program is less than 1msec**

**Test Case 2 :-**

**A → I1, I2, I3, I4,I5, I6,I7**

**B → I1, I2, I3, I4,I5, I6,I7**

**C → I1, I2, I3, I4,I5, I6,I7**

**D → I1, I2, I3, I4,I5, I6,I7**

**E→ I1, I2, I3, I4,I5, I6,I7**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E→ I5**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E→ I6**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E→ I7**

**A → I1**

**B → I2**

**C → I3**

**D → I5**

**E→ I4**

**Time taken by program is : 1.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E→ I5**

**A → I5**

**B → I4**

**C → I3**

**D → I2**

**E→ I1**

**B → I2**

**C → I3**

**D → I4**

**E→ I5**

**B → I4**

**C → I3**

**D → I2**

**E→ I1**

**total time taken by program 0.000000 sec**

**Test Case 3:-**

**A → I1, I2, I3, I4,I5, I6,I7**

**B → I1, I2, I3, I4,I5, I6,I7**

**D → I1, I2, I3, I4,I5, I6,I7**

**E → I1, I2, I3, I4,I5, I6,I7**

**F → I1, I2, I3, I4,I5, I6,I7**

**G → I1, I2, I3, I4,I5, I6,I7**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I7**

**G → I6**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I6**

**F → I5**

**G → I7**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I6**

**F → I7**

**G → I5**

**Time taken by program is : 26.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**A → I7**

**B → I6**

**C → I5**

**D → I4**

**E → I3**

**F → I2**

**G → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**B → I6**

**C → I5**

**D → I4**

**E → I3**

**F → I2**

**G → I1**

**total time taken by program less than 1 msec**

**Test Case 4:-**

**A → I1, I2, I3, I4,I5, I6**

**B → I1, I2, I3, I4,I5, I6**

**D → I1, I2, I3, I4,I5, I6**

**E → I1, I2, I3, I4,I5, I6**

**F → I1, I2, I3, I4,I5, I6**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I6**

**F → I5**

**A → I1**

**B → I2**

**C → I3**

**D → I5**

**E → I4**

**F → I6**

**A → I1**

**B → I2**

**C → I3**

**D → I5**

**E → I6**

**F → I4**

**Time taken by program is : 1.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**A → I6**

**B → I5**

**C → I4**

**D → I3**

**E → I2**

**F → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**B → I5**

**C → I4**

**D → I3**

**E → I2**

**F → I1**

**total time taken by program is less than 1 msec**

**Test Case 5:-**

**A → I1, I2, I3, I4,I5, I6,I7,I8**

**B → I1, I2, I3, I4,I5, I6,I7,I8**

**C → I1, I2, I3, I4,I5, I6,I7,I8**

**D → I1, I2, I3, I4,I5, I6,I7,I8**

**E → I1, I2, I3, I4,I5, I6,I7,I8**

**F → I1, I2, I3, I4,I5, I6,I7,I8**

**G → I1, I2, I3, I4,I5, I6,I7,I8**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I8**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I7**

**G → I6**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I7**

**G → I8**

**Time taken by program is : 475.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**A → I7**

**B → I6**

**C → I5**

**D → I4**

**E → I3**

**F → I2**

**G → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**B → I6**

**C → I5**

**D → I4**

**E → I3**

**F → I2**

**G → I1**

**total time taken by program is less than 1m sec**

**Test Case 6:-**

**A → I2, I3, I4,I5, I6,I7**

**B → I2, I3, I4,I5, I6,I7**

**C → I2, I3, I4,I5, I6,I7**

**D → I2, I3, I4,I5, I6,I7**

**E → I2, I3, I4,I5, I6,I7**

**F → I2, I3, I4,I5, I6,I7**

**G → I2, I3, I4,I5, I6,I7**

**Using 1 Algorithm :-**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I7**

**G → I6**

**B → I2**

**C → I3**

**D → I4**

**E → I6**

**F → I5**

**G → I7**

**B → I2**

**C → I3**

**D → I4**

**E → I6**

**F → I7**

**G → I5**

**Time taken by program is : 7.000000 sec**

**Using 2 Algorithm :-**

**A → I2**

**B → I3**

**C → I4**

**D → I6**

**E → I5**

**F → I7**

**B → I7**

**C → I6**

**D → I5**

**E → I4**

**F → I3**

**G → I2**

**A → I2**

**B → I3**

**C → I4**

**D → I6**

**E → I5**

**C → I6**

**D → I5**

**E → I4**

**F → I3**

**G → I2**

**total time taken by program less than 1 msec**

**Test Case 7:-**

**A → I1, I2, I3, I4,I5**

**B → I1, I2, I3, I4,I5**

**C →I1, I2, I3, I4,I5**

**D → I1, I2, I3, I4,I5**

**E → I1, I2, I3, I4,I5**

**F → I1, I2, I3, I4,I5**

**G → I1, I2, I3, I4,I5**

**H →I1, I2, I3, I4,I5**

**Using 1 Algorithm :-**

**D → I1**

**E → I2**

**F → I3**

**G → I4**

**H → I5**

**D → I1**

**E → I2**

**F → I3**

**G → I5**

**H → I4**

**D → I1**

**E → I2**

**F → I4**

**G → I3**

**H → I5**

**D → I1**

**E → I2**

**F → I4**

**G → I5**

**H → I3**

**Time taken by program is : 3.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**D → I5**

**E → I4**

**F → I3**

**G → I2**

**H → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I4**

**F → I3**

**G → I2**

**H → I1**

**total time taken by program is less than 1 msec**

**Test Case 8:-**

**A → I1, I2, I3, I4,I5**

**B → I1, I2, I3, I4,I5**

**D → I1, I2, I3, I4,I5**

**E → I1, I2, I3, I4,I5**

**Using 1 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**A → I1**

**B → I2**

**C → I3**

**D → I5**

**E → I4**

**A → I1**

**B → I2**

**C → I4**

**D → I3**

**E → I5**

**A → I1**

**B → I2**

**C → I4**

**D → I5**

**E → I3**

**Time taken by program is : 3.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**A → I5**

**B → I4**

**C → I3**

**D → I2**

**E → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**B → I4**

**C → I3**

**D → I2**

**E → I1**

**total time taken by program less than 1 msec**

**Test Case 9:-**

**A → I2, I3, I4,I5,I6,I9**

**B → I2, I3, I4,I5,I6,I9**

**C → I2, I3, I4,I5,I6,I9**

**D → I2, I3, I4,I5,I6,I9**

**E → I2, I3, I4,I5,I6,I9**

**Using 1 Algorithm :-**

**A → I2**

**B → I3**

**C → I4**

**D → I5**

**E → I6**

**A → I2**

**B → I3**

**C → I4**

**D → I5**

**E → I9**

**A → I2**

**B → I3**

**C → I4**

**D → I6**

**E → I5**

**A → I1**

**B → I2**

**C → I3**

**D → I6**

**E → I9**

**Time taken by program is less than 1 msec**

**Using 2 Algorithm :-**

**A → I2**

**B → I3**

**C → I4**

**D → I5**

**E → I6**

**A → I6**

**B → I5**

**C → I4**

**D → I3**

**E → I2**

**A → I2**

**B → I3**

**C → I4**

**D → I5**

**B → I5**

**C → I4**

**D → I3**

**E → I2**

**total time taken by program less than 1 msec**

**Test Case 10:-**

**A → I1,I3, I4,I5,I6,I7**

**B → I1,I3, I4,I5,I6,I7**

**C → I1,I3, I4,I5,I6,I7**

**D → I1,I3, I4,I5,I6,I7**

**E → I1,I3, I4,I5,I6,I7**

**F → I1,I3, I4,I5,I6,I7**

**G → I1,I3, I4,I5,I6,I7**

**H → I1,I3, I4,I5,I6,I7**

**I → I1,I3, I4,I5,I6,I7**

**Using 1 Algorithm :-**

**D → I1**

**E → I3**

**F → I4**

**G → I5**

**H → I6**

**I → I7**

**D → I1**

**E → I3**

**F → I4**

**G → I5**

**H → I7**

**I → I6**

**D → I1**

**E → I3**

**F → I4**

**G → I6**

**H → I5**

**I → I7**

**D → I1**

**E → I3**

**F → I4**

**G → I6**

**H → I7**

**I → I5**

**Time taken by program is : 338.000000 sec**

**Using 2 Algorithm :-**

**A → I1**

**B → I3**

**C → I4**

**D → I5**

**E → I6**

**F → I7**

**D → I7**

**E → I6**

**F → I5**

**G → I4**

**H → I3**

**I → I1**

**A → I1**

**B → I3**

**C → I4**

**D → I5**

**E → I6**

**E → I6**

**F → I5**

**G → I4**

**H → I3**

**I → I1**

**total time taken by program less than 1 msec**

**Test Case 11:-**

**A → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**B → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**C → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**D → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**E → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**F → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**G → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**H → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**I → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**J → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**K → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**L → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**M → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**N → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**O → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**P → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**Q → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**R → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**S → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**T → I1,I2,I3, I4,I5,I6,I7,I8,I9,I10,I11,I12,I3,I14,I15,I16,I17,I18,I19,I20**

**Using 1 Algorithm :-**

**Algorithm Fails**

**Using 2 Algorithm :-**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**H → I8**

**I → I9**

**J → I10**

**K → I11**

**L → I12**

**M → I13**

**N → I14**

**O → I15**

**P → I16**

**Q → I17**

**R → I18**

**S → I19**

**T → I20**

**A → I20**

**B → I19**

**C → I18**

**D → I17**

**E → I16**

**F → I15**

**G → I14**

**H → I13**

**I → I12**

**J → I11**

**K → I10**

**L → I9**

**M → I8**

**N → I7**

**O → I6**

**P → I5**

**Q → I4**

**R → I3**

**S → I2**

**T → I1**

**A → I1**

**B → I2**

**C → I3**

**D → I4**

**E → I5**

**F → I6**

**G → I7**

**H → I8**

**I → I9**

**J → I10**

**K → I11**

**L → I12**

**M → I13**

**N → I14**

**O → I15**

**P → I16**

**Q → I17**

**R → I18**

**S → I19**

**B → I19**

**C → I18**

**D → I17**

**E → I16**

**F → I15**

**G → I14**

**H → I13**

**I → I12**

**J → I11**

**K → I10**

**L → I9**

**M → I8**

**N → I7**

**O → I6**

**P → I5**

**Q → I4**

**R → I3**

**S → I2**

**T → I1**

**total time taken by program is less than 1msec**