```
1)a) import java.util.Scanner;     class Student
{ String USN, Name, Branch, Phone;
Scanner input = new Scanner(System.in);
void read( ) {   System.out.println("Enter Student
Details");   System.out.println("Enter USN");
USN = input.nextLine();   System.out.println("Enter
Name");   Name = input.nextLine();
System.out.println("Enter Branch");
Branch = input.nextLine();   System.out.println("Enter
Phone");   Phone = input.nextLine();  }
void display( )
{  System.out.printf("%-20s %-20s %-20s %-20s",
USN, Name, Branch, Phone);  }   }
class studentdetails
{  public static void main(String[ ] args)
{   Scanner input = new Scanner(System.in);
System.out.println("Enter number of student details to
be created");   int number = input.nextInt( );
Student s[ ] = new Student[number];
for (int i = 0; i < number; i++)
{   s[i] = new Student( );   s[i].read( );  }
System.out.printf("%-20s %-20s %-20s %-20s", "USN",
"NAME", "BRANCH", "PHONE");
for (int i = 0; i < number; i++)   {  System.out.println( );
s[i].display( );   }     input.close( );  }  }
```

```
1)b)import java.util.*;  class arrayStack
{  int arr[ ];  int top, max;
arrayStack(int n)
{  max = n;   arr = new int[max];  top = -1;  }
void push(int i)
{  if (top == max - 1)
System.out.println("Stack Overflow");
else
arr[++top] = i;  }
void pop( )
{  if (top == -1)  {  System.out.println("Stack
Underflow");  }
else     {  int element = arr[top--];
System.out.println("Popped Element: " + element);
}  }    void display( )
{  System.out.print("\nStack = ");
if (top == -1)
{  System.out.print("Empty\n");   return;  }
for (int i = top; i >= 0; i--)
System.out.print(arr[i] + " ");  System.out.println( );  } }
class Stack
{  public static void main(String[ ] args)
{  Scanner scan = new Scanner(System.in);
System.out.println("Enter Size of Integer Stack ");
int n = scan.nextInt( );  boolean done = false;
arrayStack stk = new arrayStack(n);  char ch;
do
{ System.out.println("\nStack Operations");
System.out.println("1. push");  System.out.println("2.
pop");  System.out.println("3. display");
System.out.println("4. Exit");
int choice = scan.nextInt( );
switch (choice)
{ case 1:
System.out.println("Enter integer element to push");
stk.push(scan.nextInt( ));  break;
case 2:
stk.pop( );  break;
case 3:
stk.display( );  break;
case 4:
done = true;  break;
default:
System.out.println("Wrong Entry \n ");  break;  }  }
while (!done);  }  }
```

```
2)a)import java.util.Scanner;   class Staff
{  String StaffID, Name, Phone, Salary;
Scanner input = new Scanner(System.in);
void read( )
{  System.out.println("Enter StaffID");
StaffID = input.nextLine( );
System.out.println("Enter Name");
Name = input.nextLine( );
System.out.println("Enter Phone");
Phone = input.nextLine( );
System.out.println("Enter Salary");
Salary = input.nextLine( );  }
void display( )
{ System.out.printf("\n%-15s", "STAFFID: ");
System.out.printf("%-15s \n", StaffID);
System.out.printf("%-15s", "NAME: ");
System.out.printf("%-15s \n", Name);
System.out.printf("%-15s", "PHONE:");
System.out.printf("%-15s \n", Phone);
System.out.printf("%-15s", "SALARY:");
System.out.printf("%-15s \n", Salary); }  }
class Teaching extends Staff
{ String Domain, Publication;
void read_Teaching( )
{  super.read( ); System.out.println("Enter Domain");
Domain = input.nextLine( );  System.out.println("Enter
Publication");  Publication = input.nextLine( );
} void display( )
{ super.display( );  System.out.printf("%-15s",
"DOMAIN:");  System.out.printf("%-15s \n", Domain);
System.out.printf("%-15s", "PUBLICATION:");
System.out.printf("%-15s \n", Publication);  }  }
class Technical extends Staff
{  String Skills;
void read_Technical( )
{ super.read( );  System.out.println("Enter Skills");
Skills = input.nextLine( );  }
void display( )
{  super.display( );  System.out.printf("%-15s",
"SKILLS:");  System.out.printf("%-15s \n", Skills);  }  }
```

```
class Contract extends Staff
{ String Period;   void read_Contract( )
{ super.read( );  System.out.println("Enter Period");
Period = input.nextLine( );  }
void display( )
{ super.display( );   System.out.printf("%-15s",
"PERIOD:");  System.out.printf("%-15s \n", Period);  }}
class Staffdetails
{ public static void main(String[ ] args)
{ Scanner input = new Scanner(System.in);
System.out.println("Enter number of staff details to be
created");   int n = input.nextInt( );
Teaching steach[] = new Teaching[n];  Technical stech[]
= new Technical[n];  Contract scon[] = new Contract[n];
for (int i = 0; i < n; i++)
{  System.out.println("Enter Teaching staff
information");  steach[i] = new Teaching( );
steach[i].read_Teaching ( );  }
for (int i = 0; i < n; i++)
{ System.out.println("Enter Technical staff
information");  stech[i] = new Technical( );
stech[i].read_Technical( );  }
for (int i = 0; i < n; i++)
{  System.out.println("Enter Contract staff
information");  scon[i] = new Contract( );
scon[i].read_Contract( );  }
System.out.println("\n STAFF DETAILS: \n");
System.out.println("-----TEACHING STAFF
DETAILS----- ");
for (int i = 0; i < n; i++)
{ steach[i].display( );  }
System.out.println( );  System.out.println("-----
TECHNICAL STAFF DETAILS-----");
for (int i = 0; i < n; i++)
{ stech[i].display( );  }
System.out.println( );  System.out.println("-----
CONTRACT STAFF DETAILS-----");
for (int i = 0; i < n; i++)
{  scon[i].display( );  }  input.close();  }  }
```

```
2)b)import java.util.Scanner;
import java.util.StringTokenizer;
public class Customer
{  public static void main(String[ ] args)
{  String name; Scanner scan = new Scanner(System.in);
System.out.println("Enter Name and Date_of_Birth in
the format Name,DD/MM/YYYY>");
name = scan.next( );
StringTokenizer st = new StringTokenizer(name, ",/");
int count = st.countTokens( );
for (int i = 1; i <= count && st.hasMoreTokens( ); i++)
{ System.out.print(st.nextToken( ));
if (i < count)
System.out.print(",");  }  }  }
```

```
3)a)import java.util.Scanner;
class exception
{  public static void main(String[ ] args)
{ int a, b, result;
Scanner input = new Scanner(System.in);
System.out.println("Input two integers");
a = input.nextInt( );  b = input.nextInt( );
try
{ result = a / b;  System.out.println("Result = " + result);
}  catch (ArithmeticException e)
{ System.out.println("Exception caught: Division by zero.");  }  }  }
```

```
3)b)import java.util.Random;
class SquareThread implements Runnable
{ int x;    SquareThread(int x)  {  this.x = x; }
public void run( )
{ System.out.println("Thread Name:Square Thread and
Square of " + x + " is: " + x * x);  }  }
class CubeThread implements Runnable
{  int x;  CubeThread(int x)  { this.x = x; }
public void run( )
{ System.out.println("Thread Name:Cube Thread and
Cube of " + x + " is: " + x * x * x); } }
class RandomThread implements Runnable
{ Random r; Thread t2, t3;
public void run( )
{ int num;  r = new Random( );
```

```
try    { while (true)
{ num = r.nextInt(100);
System.out.println("Main Thread and Generated
Number is " + num);  t2 = new Thread(new
SquareThread(num));  t2.start( );
t3 = new Thread(new CubeThread(num));  t3.start( );
Thread.sleep(1000); System.out.println("------------------
--------------------");  }  }
catch (Exception ex)
{ System.out.println("Interrupted Exception"); } } }
public class MainThread
{ public static void main(String[ ] args)
{ RandomThread thread_obj = new RandomThread( );
Thread t1 = new Thread(thread_obj); t1.start( ); }  }
```

```
4)import java.util.Scanner;
import java.util.Arrays;
import java.util.Random;
public class QuickSortComplexity
{ static final int MAX = 10005;
static int[] a = new int[MAX];
public static void main(String[ ] args)
{ Scanner input = new Scanner(System.in);
System.out.print("Enter Max array size: ");
int n = input.nextInt();
Random random = new Random();
// System.out.println("Enter the array elements: ");
for (int i = 0; i < n; i++)
// a[i] = input.nextInt(); // for keyboard entry
a[i] = random.nextInt(1000); // generate  random
a = Arrays.copyOf(a, n); // keep only non zero elements
// QuickSortAlgorithm(0, n - 1);// for worst-case time
complexity
// System.out.println("Input Array:");
// for (int i = 0; i < n; i++)
// System.out.print(a[i] + " ");
// set start time
long startTime = System.nanoTime( );
QuickSortAlgorithm(0, n - 1);
long stopTime = System.nanoTime( );
long elapsedTime = stopTime - startTime;
/* System.out.println("\nSorted Array:");
for (int i = 0; i < n; i++)
System.out.print(a[i] + " ");
System.out.println( ); */
System.out.println("Time Complexity in ms for
n=" + n + " is: " + (double) elapsedTime / 1000000);  }
public static void QuickSortAlgorithm(int p, int r)
{ int i, j, temp, pivot;
if (p < r)
{ i = p;  j = r + 1;
pivot = a[p]; // mark first element as pivot
while (true)   { i++;
while (a[i] < pivot && i < r)
i++;  j--;
while (a[j] > pivot)
j--;
if (i < j)
{ temp = a[i]; a[i] = a[j]; a[j] = temp;  }
else
break; // partition is over } a[p] = a[j]; a[j] = pivot;
QuickSortAlgorithm(p, j - 1);
QuickSortAlgorithm(j + 1, r); } } }
```

5)
```
import java.util.Random;
import java.util.Scanner;
public class MergeSort2
{ static final int MAX = 10005;
static int[] a = new int[MAX];
public static void main(String[ ] args)
{ Scanner input = new Scanner(System.in);
System.out.print("Enter Max array size: ");
int n = input.nextInt( );
Random random = new Random( );
// System.out.println("Enter the array elements: ");
for (int i = 0; i < n; i++)
//a[i] = input.nextInt(); // for keyboard entry
a[i] = random.nextInt(1000); // generate random
// MergeSortAlgorithm(0, n - 1);
long startTime = System.nanoTime();
MergeSortAlgorithm(0, n - 1);
long stopTime = System.nanoTime();
long elapsedTime = stopTime - startTime;
System.out.println("Time Complexity (ms) for n = " +
n + " is : " + (double) elapsedTime / 1000000);
// System.out.println("Sorted Array (Merge Sort):");
// for (int i = 0; i < n; i++)
// System.out.print(a[i] + " "); input.close(); }
public static void MergeSortAlgorithm(int low, int high)
{ int mid;
if (low < high)
{ mid = (low + high) / 2; MergeSortAlgorithm(low,
mid); MergeSortAlgorithm(mid + 1, high); Merge(low,
mid, high); } }
j = mid + 1;
while ((h <= mid) && (j <= high))
```

```
public static void Merge(int low, int mid, int high)
{ int[ ] b = new int[MAX]; int i, h, j, k; h = i = low;
if (a[h] < a[j])
b[i++] = a[h++];
else          b[i++] = a[j++];
if (h > mid)
for (k = j; k <= high; k++)
b[i++] = a[k];
else          for (k = h; k <= mid; k++)
b[i++] = a[k];
for (k = low; k <= high; k++)
a[k] = b[k]; }  }
```

```
6)a)import java.util.Scanner;
public class KnapsackDP
{ static final int MAX = 20; static int w[ ];
static int p[ ]; static int n; static int M;
static int V[ ][ ]; static int Keep[ ] [ ];
public static void main(String args[ ])
{ w = new int[MAX]; p = new int[MAX];
V = new int [MAX][MAX];
Keep = new int[MAX][MAX]; int optsoln;
ReadObjects( );
for (int i = 0; i <= M; i++)
V[0][i] = 0;
for (int i = 0; i <= n; i++)
V[i][0] = 0; optsoln = Knapsack( );
System.out.println("Optimal solution = " + optsoln); }
static int Knapsack( )
{ int r;   for (int i = 1; i <= n; i++)
for (int j = 0; j <= M; j++)
if ((w[i] <= j) && (p[i] + V[i - 1][j - w[i]] > V[i - 1][j]))
{ V[i][j] = p[i] + V[i - 1][j - w[i]];  Keep[i][j] = 1;  }
```

```
else
{ V[i][j] = V[i - 1][j];   Keep[i][j] = 0;  }
r = M; System.out.println("Items = ");
for (int i = n; i > 0; i--)
if (Keep[i][r] == 1)
{ System.out.println(i + " "); r = r - w[i]; }
System.out.println( ); return V[n][M]; }
static void ReadObjects( )
{ Scanner scanner = new Scanner(System.in);
System.out.println("Knapsack Problem - Dynamic
Programming Solution: ");  System.out.println("Enter
the max capacity of knapsack: ");
M = scanner.nextInt( ); System.out.println("Enter
number of objects: "); n = scanner.nextInt( );
System.out.println("Enter Weights: ");
for (int i = 1; i <= n; i++)
w[i] = scanner.nextInt( );
System.out.println("Enter Profits: ");
for (int i = 1; i <= n; i++)
p[i] = scanner.nextInt( ); scanner.close( ); }  }
```

```
6)b) import java.util.Scanner;
class KObject
{  float w; float p; float r;  }
public class KnapsackGreedy2
{ static final int MAX = 20;
static int n; static float M;
public static void main(String args[ ])
{ Scanner scanner = new Scanner(System.in);
System.out.println("Enter number of objects: ");
n = scanner.nextInt( ); KObject[ ] obj = new
KObject[n];     for(int i = 0; i<n;i++)
obj[i] = new KObject( );// allocate memory for members
ReadObjects(obj); Knapsack(obj); scanner.close();  }
static void ReadObjects(KObject obj[ ])
{ KObject temp = new KObject( );
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the max capacity of knapsack:
");  M = scanner.nextFloat( );
System.out.println("Enter Weights: ");
for (int i = 0; i < n; i++)
obj[i].w = scanner.nextFloat( );
```

```
System.out.println("Enter Profits: ");
for (int i = 0; i < n; i++)
obj[i].p = scanner.nextFloat( );
for (int i = 0; i < n; i++)
obj[i].r = obj[i].p / obj[i].w;
for(int i = 0; i<n-1; i++)  for(int j=0; j<n-1-i; j++)
if(obj[j].r < obj[j+1].r)
{  temp = obj[j]; obj[j] = obj[j+1]; obj[j+1] = temp;  }
scanner.close( );  }
static void Knapsack(KObject kobj[ ])
{ float x[ ] = new float[MAX]; float totalprofit;
int i; float U; U = M; totalprofit = 0;
```

```
for (i = 0; i < n; i++)        x[i] = 0;
for (i = 0; i < n; i++)     { if (kobj[i].w > U)    break;
else {  x[i] = 1;
totalprofit = totalprofit + kobj[i].p;
U = U - kobj[i].w;  }  }
System.out.println("i = " + i);
if (i < n)       x[i] = U / kobj[i].w;
totalprofit = totalprofit + (x[i] * kobj[i].p);
System.out.println("The Solution vector, x[ ]: ");
for (i = 0; i < n; i++)
System.out.print(x[i] + " "); System.out.println("\nTotal
profit is = " + totalprofit); }  }
```

```
7)import java.util.*;
public class DijkstrasClass
{ final static int MAX = 20;
final static int infinity = 9999;
static int n; static int a[ ][ ];
static Scanner scan = new Scanner(System.in);
public static void main(String[ ] args)
{ ReadMatrix( ); int s = 0;
System.out.println("Enter starting vertex: ");
s = scan.nextInt( ); Dijkstras(s); }
static void ReadMatrix( )
{ a = new int[MAX][MAX]; System.out.println("Enter
the number of vertices:"); n = scan.nextInt( );
System.out.println("Enter the cost adjacency matrix:");
for (int i = 1; i <= n; i++)   for (int j = 1; j <= n; j++)
a[i][j] = scan.nextInt( ); }
static void Dijkstras(int s)
```

```
{ int S[ ] = new int[MAX];  int d[ ] = new int[MAX];
int u, v; int i;     for (i = 1; i <= n; i++)
{ S[i] = 0;  d[i] = a[s][i]; }
S[s] = 1; d[s] = 1;  i = 2;
while (i <= n)
{ u = Extract_Min(S, d);   S[u] = 1;   i++;
for (v = 1; v <= n; v++)
{ if (((d[u] + a[u][v] < d[v]) && (S[v] == 0)))
d[v] = d[u] + a[u][v]; } }
for (i = 1; i <= n; i++)
if (i != s)
System.out.println(i + ":" + d[i]); }
static int Extract_Min(int S[ ], int d[ ])
{ int i, j = 1, min;  min = infinity;
for (i = 1; i <= n; i++)
{  if ((d[i] < min) && (S[i] == 0))
{ min = d[i];  j = i;  }  }  return (j);  }  }
```

```
8)import java.util.Scanner;
public class KruskalsClass
{ final static int MAX = 20;
static int n; static int cost[ ][ ];
static Scanner scan = new Scanner(System.in);
public static void main(String[ ] args)
{ ReadMatrix( ); Kruskals( ); }
static void ReadMatrix( )
{ int i, j; cost = new int[MAX][MAX];
System.out.println("Implementation of Kruskal's
algorithm");  System.out.println("Enter the no. of
vertices");  n = scan.nextInt( );
System.out.println("Enter the cost adjacency matrix");
for (i = 1; i <= n; i++)
{ for (j = 1; j <= n; j++) { cost[i][j] = scan.nextInt( );
if (cost[i][j] == 0)    cost[i][j] = 999; }  }  }
static void Kruskals( )
{ int a = 0, b = 0, u = 0, v = 0, i, j, ne = 1, min, mincost
= 0;   System.out.println("The edges of Minimum Cost
Spanning Tree are");
while (ne < n)
{ for (i = 1, min = 999; i <= n; i++) {
for (j = 1; j <= n; j++)
{ if (cost[i][j] < min)
{ min = cost[i][j];  a = u = i;  b = v = j;  } } }
u = find(u);  v = find(v);
if (u != v)
{ uni(u, v); System.out.println(ne++ + "edge (" + a + ","
+ b + ") =" + min);  mincost += min;  }
cost[a][b] = cost[b][a] = 999; }
System.out.println("Minimum cost :" + mincost);  }
static int find(int i)
{ int parent[ ] = new int[9];
while (parent[i] == 1)
i = parent[i];  return i;  }
static void uni(int i, int j)
{ int parent[ ] = new int[9];  parent[j] = i;  }  }
```

```
9)import java.util.Scanner;
public class PrimsClass
{ final static int MAX = 20;
static int n; static int cost[ ][ ];
static Scanner scan = new Scanner(System.in);
public static void main(String[ ] args)
{ ReadMatrix( );  Prims( ); }
static void ReadMatrix( )
{ int i, j; cost = new int[MAX][MAX];
System.out.println("\n Enter the number of nodes:");
n = scan.nextInt( ); System.out.println("\n Enter the
adjacency matrix:\n");
for (i = 1; i <= n; i++)    for (j = 1; j <= n; j++)
{ cost[i][j] = scan.nextInt( );
if (cost[i][j] == 0)
cost[i][j] = 999; }  }
static void Prims( )
{ int visited[ ] = new int[10];
int ne = 1, i, j, min, a = 0, b = 0, u = 0, v = 0;
int mincost = 0; visited[1] = 1;
while (ne < n)
{ for (i = 1, min = 999; i <= n; i++)
for (j = 1; j <= n; j++)   if (cost[i][j] < min)
if (visited[i] != 0)
{ min = cost[i][j];   a = u = i; b = v = j; }
if (visited[u] == 0 || visited[v] == 0)
{ System.out.println("Edge" + ne++ + ":(" + a + "," + b
+ ")" + "cost:" + min);   mincost += min;  visited[b] = 1;
} cost[a][b] = cost[b][a] = 999; }
System.out.println("\n Minimum cost" + mincost); }  }
```

```
10)a) import java.util.Scanner;  public class FloydsClass
{ static final int MAX = 20; static int a[ ][ ];  static int n;
public static void main(String args[ ])
{ a = new int[MAX][MAX];  ReadMatrix( );
Floyds( ); PrintMatrix( );  }
static void ReadMatrix( )
{  System.out.println("Enter the number of vertices\n");
Scanner scanner = new Scanner(System.in);
n = scanner.nextInt( ); System.out.println("Enter the
Cost Matrix (999 for infinity) \n");
for (int i = 1; i <= n; i++)  { for (int j = 1; j <= n; j++)
{ a[i][j] = scanner.nextInt( ); } }  scanner.close( );  }
static void Floyds( )
{ for (int k = 1; k <= n; k++) { for (int i = 1; i <= n; i++)
for (int j = 1; j <= n; j++)
if ((a[i][k] + a[k][j]) < a[i][j])
a[i][j] = a[i][k] + a[k][j]; }  }
static void PrintMatrix( )
{ System.out.println("The All Pair Shortest Path Matrix
is:\n");        for(int i=1; i<=n; i++)
{ for(int j=1; j<=n; j++)    System.out.print(a[i][j] +
"\t"); System.out.println("\n"); }  }  }
```

```
10)b) import java.util.Scanner;
public class TravSalesPerson
{ static int MAX = 100; static final int infinity = 999;
public static void main(String args[ ])
{ int cost = infinity; int c[ ][ ] = new int[MAX][MAX];
int tour[ ] = new int[MAX];
int n; System.out.println("Travelling Salesman Problem
using Dynamic Programming\n");
System.out.println("Enter number of cities: ");
Scanner scanner = new Scanner(System.in);
n = scanner.nextInt( ); System.out.println("Enter Cost
matrix:\n");     for (int i = 0; i < n; i++)
for (int j = 0; j < n; j++)
{ c[i][j] = scanner.nextInt( );
if (c[i][j] == 0)       c[i][j] = 999; }
for (int i = 0; i < n; i++)
tour[i] = i; cost = tspdp(c, tour, 0, n);
System.out.println("Minimum Tour Cost: " + cost);
System.out.println("\nTour:");
```

```
for (int i = 0; i < n; i++)
{ System.out.print(tour[i] + " -> "); }
System.out.println(tour[0] + "\n"); scanner.close( ); }
static int tspdp(int c[ ][ ], int tour[ ], int start, int n)
{ int i, j, k; int temp[ ] = new int[MAX];
int mintour[ ] = new int[MAX]; int mincost;  int cost;
if (start == n - 2)
 return c[tour[n - 2]][tour[n - 1]] + c[tour[n - 1]][0];
mincost = infinity;
for (i = start + 1; i < n; i++)   { for (j = 0; j < n; j++)
temp[j] = tour[j]; temp[start + 1] = tour[i];
temp[i] = tour[start + 1];
if (c[tour[start]][tour[i]] + (cost = tspdp(c, temp, start +
1, n)) < mincost) {
mincost = c[tour[start]][tour[i]] + cost;
for (k = 0; k < n; k++)
mintour[k] = temp[k]; }  }
for (i = 0; i < n; i++)
tour[i] = mintour[i];  return mincost;  }  }
```

```
11)import java.util.Scanner;  public class SumOfsubset
{ final static int MAX = 10; static int n; static int S[ ];
static int soln[ ]; static int d;
public static void main(String args[ ])
{ S = new int[MAX]; soln = new int[MAX];
int sum = 0; Scanner scanner = new Scanner(System.in);
System.out.println("Enter number of elements: ");
n = scanner.nextInt( );
System.out.println("Enter the set in increasing order: ");
for (int i = 1; i <= n; i++)
S[i] = scanner.nextInt( ); System.out.println("Enter the
max. subset value(d): ");
d = scanner.nextInt( );
for (int i = 1; i <= n; i++)
sum = sum + S[i];
if (sum < d || S[1] > d)
System.out.println("No Subset possible");
else      SumofSub(0, 0, sum); scanner.close( );  }
static void SumofSub(int i, int weight, int total)
{ if (promising(i, weight, total) == true)
if (weight == d)
{ for (int j = 1; j <= i; j++)
{ if (soln[j] == 1)
System.out.print(S[j] + " "); } System.out.println( ); }
else
{ soln[i + 1] = 1;
SumofSub(i + 1, weight + S[i + 1], total - S[i + 1]);
soln[i + 1] = 0;
SumofSub(i + 1, weight, total - S[i + 1]); } }
static boolean promising(int i, int weight, int total)
{ return ((weight + total >= d) && (weight == d ||
weight + S[i + 1] <= d)); }  }
```

```
12) import java.util.Scanner; public class Hamiltonian
{ boolean found = false; int G[ ][ ];
int x[ ]; int n;
public static void main(String args[ ])
{ Hamiltonian hamiltonian = new Hamiltonian( );
hamiltonian.getData( );
System.out.println("\nSolution:");
hamiltonian.HamiltonianMethod(2);
hamiltonian.printNoSlnPossible( ); }
public void printNoSlnPossible( )
{ if (found == false)
System.out.println("No Solution possible!"); }
public void getData( )
{ Scanner scanner = new Scanner(System.in);
System.out.println("\t\t\t\tHamiltonian Cycle");
System.out.print("\nEnter the number of the vertices: ");
n = scanner.nextInt( ); G = new int[n + 1][n + 1];
x = new int[n + 1]; System.out.print("\nIf edge between
the following vertices enter 1 else 0:\n");
for (int i = 1; i <= n; i++)
for (int j = 1; j <= n; j++)
{ if ((i != j) && (i < j))
{ System.out.print(i + " and " + j + ": ");
G[j][i] = G[i][j] = scanner.nextInt( ); }
if (i == j)       G[i][j] = 0; }
for (int i = 1; i <= n; i++)
x[i] = 0; x[1] = 1; scanner.close( ); }
void HamiltonianMethod(int k)
{ while (true) { NextValue(k, G, x, n);
if (x[k] == 0)    return;
if (k == n)    { for (int i = 1; i <= k; i++)
System.out.print(x[i] + " "); System.out.println(x[1]);
System.out.println( ); found = true; return; }
else     HamiltonianMethod(k + 1); } }
void NextValue(int k, int G[ ][ ], int x[ ], int n)
{ while (true) { x[k] = (x[k] + 1) % (n + 1);
if (x[k] == 0)        return;
if (G[x[k - 1]][x[k]] != 0) { int j;     for (j = 1; j < k; j++)
if (x[k] == x[j])    break;
if (j == k)
if ((k < n) || ((k == n) && G[x[n]][x[1]] != 0))
return; } } } }
```