Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No. 10(ii)

PROBLEM STATEMENT: Design and implement a program to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking Principle.

ALGORITHM & CODE:

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#define MAX_VERTICES 20
int graph [MAX_VERTICES][MAX_VERTICES];
int path [MAX_VERTICES];
int n;

void printSolution ()
{
  Print ("Hamiltonian Cycle : ");
  for (int i=0; i<n; i++)
  { printf ("%d ", path[i]);
  }
  Printf ("%d \n", Path [0] );
}
bool isSafe (int v, int pos)
{ if (graph [path [pos -1]][v] == 0)
  { return false;
  }
  for (int i = 0; i < pos; i++)
  { if (path [i] == v)
    {
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
        return false;
    }}
    return true;
}
bool hamCycleUtil (int pos)
  {; if (pos == n)
   {; if (graph[path [pos -1]][path [0]] == 1)
     {printSolution ();
      return true;
     }
     else {
       return false;
     }
  }
 bool foundAny = false;
 for (int v=1 ; v<n; v++)
 { if (isSafe (v, pos))
   {path [pos] = v;
    foundAny = hamCycle Util (pos+1) || foundAny;
    path [pos] = -1;
   }
 }
 return foundAny;
}
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
void findHamiltonianCycles()
{
 for(int i=0; i<n; i++)
  { path[i] = -1;
  }
  Path[0] = 0;
  Printf("Searching for Hamiltonian cycles..\n");
  if(!hamCycleUtil(1)){
      Printf("No Hamiltonian cycle exists\n");
  }
}
int main()
{ Printf('Enter the number of vertices in the graph
(max %d): ", MAX_VERTICES);
   Scanf("%d", &n);
   if(n<=0 || n> MAX_VERTICES)
   {Printf("Invalid number of vertices, must be between
1 and %d.\n", MAX_VERTICES);
    return 1;
   }
   Printf("Enter the adjacency matrix (%d x %d):\n",
n, n);
   for(int i = 0; i<n; i++)
   { for(int j=0; j<n; j++)
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty	Signature of Student
Date :	Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

```
        {scanf("%d", &graph[i][j]);
    }
}
findHamiltonianCycles();
return 0;
}
```

OUTPUT

```
Enter the number of vertices in the graph
(max 20): 5
Enter the adjacency matrix (5x5):
0 1 0 1 0
1 0 1 1 1
0 1 0 0 1
1 1 0 0 1
0 1 1 1 0
Searching for Hamiltonian cycles...
Hamiltonian cycles : 0 1 2 4 3 0
Hamiltonian cycles : 0 1 3 4 2 0
Hamiltonian cycles : 0 1 4 2 3 0
Hamiltonian cycles : 0 1 4 3 2 0
Hamiltonian cycles : 0 3 1 2 4 0
Hamiltonian cycles : 0 3 1 4 2 0
Hamiltonian cycles : 0 3 2 4 1 0
Hamiltonian cycles : 0 3 4 1 2 0
Hamiltonian cycles : 0 3 4 2 1 0
```