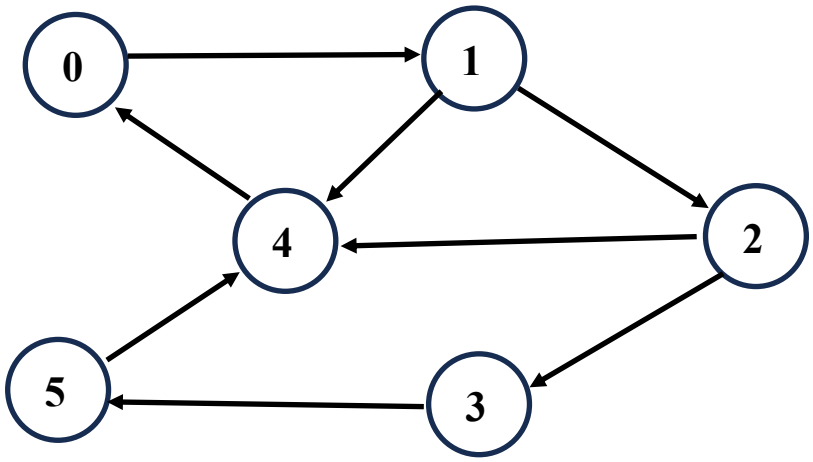
Experiment No. 08

I. Aim: Implement Hamiltonian cycles.

II. Theory:

A Hamiltonian Cycle or Circuit is a path in a graph that visits every vertex exactly once and returns to the starting vertex, forming a closed loop. A graph is said to be a Hamiltonian graph only when it contains a Hamiltonian cycle, otherwise, it is called non-Hamiltonian graph.

Give adjacency matrix for given graph. Also find Hamiltonian cycle in the graph.



III. Program:

```
#include <stdio.h>
#define NODE 6
int graph[NODE][NODE] = {
   {0, 1, 0, 0, 0, 0},
   {0, 0, 1, 0, 1, 0},
   {0, 0, 0, 1, 0, 0},
   {0, 0, 0, 0, 0, 1},
   {1, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 1, 0},
};
int path[NODE];

void displayCycle()
{
   printf("Cycle Found: ");
   for (int i = 0; i < NODE; i++)
      printf("%d ", path[i]);
   printf("%d\n", path[0]);
}
int isValid(int v, int k)
{
   if (graph[path[k - 1]][v] == 0)
```

```
    return 0;
  for (int i = 0; i < k; i++)
    if (path[i] == v)
      return 0;
  return 1;
}
int cycleFound(int k)
{
  if (k == NODE)
  {
    if (graph[path[k - 1]][path[0]] == 1)
      return 1;
    else
      return 0;
  }
  for (int v = 1; v < NODE; v++)
  {
    if (isValid(v, k))
    {
      path[k] = v;
      if (cycleFound(k + 1) == 1)
        return 1;
      path[k] = -1;
    }
  }
  return 0;
}
int hamiltonianCycle()
{
  for (int i = 0; i < NODE; i++)
    path[i] = -1;
  path[0] = 0;
  if (cycleFound(1) == 0)
  {
    printf("Solution does not exist\n");
    return 0;
  }
  displayCycle();
  return 1;
}
void main()
{
  hamiltonianCycle();
  getch();
}
```

IV. Output:

V. Conclusion: Successfully implemented code for finding Hamiltonian cycle in given graph.