

MODEL LAB EXAMINATION

NAME:THILLAI NATHAN B

ROLL NO:231501173

SEC :AIML - C

SUBJECT:POAI

CODING :

```
from collections import deque
def BFS(a, b, target):
    m = {}
    isSolvable = False
    path = []
    q = deque()
    q.append((0, 0))
    while (len(q) > 0):
        u = q.popleft()
        if ((u[0], u[1]) in m):
            continue
        if ((u[0] > a or u[1] > b or
            u[0] < 0 or u[1] < 0)):
            continue
        path.append([u[0], u[1]])
        m[(u[0], u[1])] = 1
        if (u[0] == target or u[1] == target):
            isSolvable = True
            if (u[0] == target):
                if (u[1] != 0):
                    path.append([u[0], 0])
            else:
                if (u[1] != 0):
                    path.append([0, u[1]])
            sz = len(path)
            for i in range(sz):
                print("(" + path[i][0] + ", " +
                    path[i][1] + ")")
            break
        q.append([u[0], b])
        q.append([a, u[1]])
    for ap in range(max(a, b) + 1):
        c = u[0] + ap
        d = u[1] - ap
```

```

        if (c == a or (d == 0 and d >= 0)):
            q.append([c, d])
        c = u[0] - ap
        d = u[1] + ap
        if ((c == 0 and c >= 0) or d == b):
            q.append([c, d])
        q.append([a, 0])
        q.append([0, b])
    if (not isSolvable):
        print("No solution")
if __name__ == '__main__':
    Jug1, Jug2, target = 4, 3, 2
    print("Path from initial state "
          "to solution state ::")
    BFS(Jug1, Jug2, target)

```

OUTPUT :

```

Python IDE
main.py
1 from collections import deque
2 def BFS(a, b, target):
3     m = 0
4     isSolvable = False
5     path = []
6     q = deque()
7     q.append((0, 0))
8     while (len(q) > 0):
9         u = q.popleft()
10        if ((u[0] == a or u[1] == b) or
11            (u[0] < 0 or u[1] < 0)):
12            continue
13        path.append(u[0], u[1])
14        m(u[0], u[1]) == 1
15        if (u[0] == target or u[1] == target):
16            isSolvable = True
17            if (u[0] == 0):
18                path.append((u[0], 0))
19            else:
20                if (u[1] == 0):
21                    path.append((0, u[1]))
22                else:
23                    path.append((u[0], u[1]))
24            m = len(path)
25            for i in range(m):
26                print("%s" % path[i][0], "%s" %
27                    path[i][1], " ")
28            break
29        q.append((u[0], 0))
30        q.append((a, u[1]))
31        for ap in range(max(a, b) + 1):
32            c = u[0] - ap
33            d = u[1] + ap
34            if (c == a or (d == 0 and d >= 0)):
35                q.append((c, d))
36            c = u[0] - ap
37            d = u[1] + ap
38            if ((c == 0 and c >= 0) or d == b):
39                q.append((c, d))
40        q.append((a, 0))
41        q.append((0, b))
42    if (not isSolvable):
43        print("No solution")
44    if __name__ == '__main__':
45        Jug1, Jug2, target = 4, 3, 2
46        print("Path from initial state "
47              "to solution state ::")
48        BFS(Jug1, Jug2, target)

```

Path from initial state to solution state ::

```

(0, 0)
(0, 3)
(4, 3)
(4, 0)
(1, 3)
(3, 3)
(4, 2)
(0, 2)

```

Code Execution Successful