Exp.3.

8 puzzle problem using iterative depth search bfs.

```python
def bfs ( src, target):

        visited_states = [ ]
        visited_states . append (src)
        arr = [ src]
        c = 0
        while arr:
                c += 1
                if arr[o] == target:
                        return True
                arr += possible_moves (arr[o], visited_states)
                arr. pop(o)
        return false.

def possible_moves ( state, visited_states):

        b = state . index (-1)
        d = [ ]

        if b+3 in range (9):
                d. append ('d')
        if b-3 in range (9):
                d. append ('u')
        if b not in [0,3,6]:
                d. append ('l')
        if b not in [2,5,8]:
                d. append ('r')

        pos_moves = [ ]
```

```python
for move in d:
    pos-moves.append(gen( state, move, b))

return [move for move in pos-moves if move
    not in visited -states]


def gen ( state, direction, blank-spot):
    temp =   state.copy()

    if direction == 'd':
        a: temp [ blank-spot +3]
        temp [ blank-spot +3] = temp[blank-spot]
        temp [ blank-spot] = a

    elif direction == 'u':
        a = temp[ blank-spot -3]
        temp [ blank-spot -3] = temp[ blank-spot]
        temp ( blank-spot) = a

    elif direction == 'l':
        a = temp[ blank-spot -1]
        temp [ blank-spot -1] = temp [ blan-spot]
        temp[ blan-spot] = a

    elif direction == 'r':
        a = temp[ blan-spot +1]
        temp[ blan + 1] = temp [ blank-spot]
        temp [ blan-spot] = a

    return temp.
```

```
# Test 1
    src = [1, 2, 3, -1, 4, 5, 6, 7, 8]
    target = [1, 2, 3, 4, 5, -1, 6, 7, 8]
    print (bfs( src, target ))
```