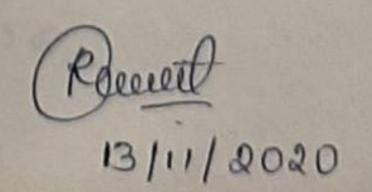
Name: Robit Kudache USN 8 1BM 18 C5083 LAB : AI LABTEST -1 Program & 8-puzzle using A* aigorithm beobeau : det process (self): # accept stort and good state print ("Enter State In") Stoot = self. accept () print (" Enter Croal State In") boal = seif. accept () Stoot = Node (Stoot, 0,0) Stant. fral = self. f (Stant, goal) # put start node in open List Deint ("10") seif. open. append (stant) while True: cut = self. open [0] for i in cur-data. tor 3 10 i: beint (7° eng = , ") point (" ") # if h value of node is 0 then we reached # goal State. : f (self. h (cwi.data, goal) == 0): break



```
(113M18CS083) AI LAB-TEST I
Robit Kudache
 Confinued ---
       for i in congenerate_child():
           i. fra1 = self. f ( is good)
             self-open-append (i)
       Self. closed. append (cur)
         del self-open [0]
     # sosting open list based on f value
     # f(n) = h(n) + g(n)
      Self. open. sort (key = lambda x: x.fval, reverse = False)
   puz = puzzle (3)
   pwz. process()
 # function f
      def f (self, start, boal):
          return Self. h (stant. data, goal) + stant. level
  # function h
        def h (self, start, goal):
      # manhattan distance absil(x2-xi) and
     # abs 142-411 or number of mispiaced tiles.
       temp = 0
       for i'n range (0, self-n):
           for j in range (0, seif.n):
              if Stoot [i] [j] )= good [i][j] and
                      Stant [1] [1] != '-' :
                    +2mp + = 1
        return temp.
```

00

```
Robit Kudache (1BM18CS083)
       def astor (stant, goal):
             States = [stant]
                0=0
             visited-stage = set ()
                 cohîle len (State):
                     Print (t" level = ( & 4")
                moves = []
             for State in States:
                   visited-State - add (tuple (State))
                       Print-prid (State)
                       18 State = 6000
                          Drint (" success")
                           return
   move + = [move for move in possible-moves
                 (State, visited - State) if more
                        not in movis]
   costs = [g+h (move, goal) for move in moves]
    States = [moves Ei] for i in ronge [len (moves) if
                      cost [i] = = min (costs)
            9+=1
       Print ("NOSOLUTION")
  # We will implement tonetion for possible
 # moves and gen.
```

Paccet 13/11/20