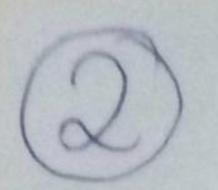
Computer Science and Engineering UG SEM~S Rutaguet Ritik Rout 1BM18CSISI 13/11/20 Course: - Artificial Intelligence Course code: - 20CS5PCAIP 2) def main (): Storting node = LCO, OJ] jugs = get - jugs () goal - amount = get - goal (jugs) check - dict = {} 15-depth = True search Esterting-node, jugs, goal-amount, check-dist, def get-index (node): return pow (7, node [0]) \* pow (5, node [1]). def get - jugs (): print ("Receiving the volume of the jugs") temp = int (input ("Enter first jug volume (>1)."))
while temp < 1. temp = int (imput ("Enter a valid amount(>1):")) jugs. append (temp) temp = int (input ("enter second jug volume(>1): "))
while temp < 1: temp = int (input ("Enter a valid amount (>1): ")) jugs. append (temp) seturn jugs

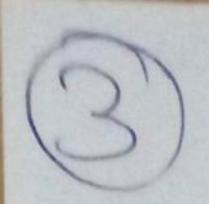
Frago



det get - goal jugs): ("retect of leaved anoual of water") max - amount - max (jusp [0], jusp [1]) war = truomo - xam . (201 - 104). - (69):" format (max comount) goal-amount = int (input(S))
while goal-amount < 1 or goal-amount > max-amount:
goal-amount = int (input ("Enter a valid amount (1-201):" format (man-amout)) return god amout. det is-god (path, god-amount): Print ("Check it bu god is achieved")
return path [-1][0] == god - amount or path [-1][1] == god - amount. det ben - there ( node, check - did):

Sint (" Check if Loy is visited before", tornat ( node)) return check - dict . get (get - index. (nodi), False) det next transitions (jusp path, that died: L J = ther unxt-rods=C] rode = [] a- max = jugs LOJ D-max = juap [1] a = path EIJ [0] b= path [-1][]

Starky.



node. append (a-max)
node. append (b)
id not been-there (node, sheck - did):
next-node. append (node)
node = EJ.

node. append (a)
node. append (b\_mar)
it not been\_there (node, check\_dict):
next\_node. append (node)
node = []

node append (min (a-mon, a+6))
node append (b-brode E0J-a))
it not been there (node, while \_dict):
next-node append chode.
node = EJ

node. append (min (a+b), b\_max).

node. insert (0), a - Cnode EOJ\_ b).

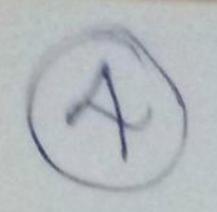
it not been - there (node, checkedict):

next\_node. append (node)

node = EJ.

node, append (b) node, append (b) is not been there (node, check dict): next - node append (node) node = []

bryang



node append (a)
node append (a)
it not been thea (node, check ded).
rext-node append (node).

ter in range (0, len (next-nades)):

temp = list (path).

temp, append (next-node [i])

result. append (temp)

it len (next\_node) == 0:

Print ("No mar & universide rods In Backtracking")

place of print ("possible transatrans:")

tor mode in next\_node:

Print (mode).

return result

det transition (old, new, jugs):

a= old [a]

b= old [1]

a-prime = new [a]

b-prime = new [1]

a-mon = jugs [0]

b-mon = jugs [1]

it as a-prime:

it b= b-prime:

return "clear fos-litter jug: 1t", format (a-max)

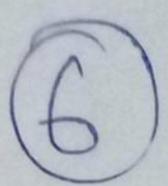
else:

return "Pour 203-litter jug into 213-litter jug: 1t" jug:

184114 6 515 5 it b> b-prime:
it a= a-prime.
return "Clear 101-WHr 14" H", found (b. 11mm) return "pour hot-Litter jugento 223- litte agé . torret (borner, a mon). it a == a prime: return "Fill 463-ditter jung "t", tormel (amy) return "Fill & Ob. Litter judy: Nt. tormed (a may) det print-potr (patrifugs). print ("Chartery brown: It" path [0]).

the im ronge (o, len (point)-1):

print (I+1) "", transition (point), point [1+1], jump), pala [1+1] def search Cstarting\_nade, jupp, god- amoust, check did, it is -gath; but the "; god D que collection. deque () a rapport left (starting - node).



while lang) != 0.

path = q. popleft ()

check -did Eget - index (path [-1]] - Irru

it last (path) > = 2.

print (transition(path [-2), path[-1), jugs), path[-1])

it is \_ god (pathy god -amount):

accompleted = True.

god = path

break

next - mones = next - transition(jugs, path, check - did)

tux i in next - mones:

it is \_ oupperd left(i).

of accomplished:

print (" the god is achied" In printing the sequence (h"),

print - path (god, jugs)

else:

print (pedat "problem cont be coloud").

it \_ name\_ = = '\_namin\_':
main c).

brisk