Department of Computer Engineering

Academic Term: Jan-Apr 2022

Class : T.E Computer Sem -VII

Subject: Artificial Intelligence

Practical No:	**8**
Title:	**Planning Problem**
Date of Performance:	**24/04/2022**
Date of Submission:	**26/04/2022**
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	Rubric	Grade
1	On time Completion & Submission(2)	
2	Output(3)	
3	Code Optimization(3)	
4	Knowledge of the topic(2)	
5	Total (10)	

Signature of the Teacher :

Practical- 08

Planning Problem

Problem Statement:

Write a program to solve the planning problem: The Block World Problem

Initial State:

- 3 blocks A, B, C
- Empty holder
- OnTable: A and C
- B is on C

Goal State:

- Empty holder
- OnTable: B
- A is on B
- C is on A

Code:

```
# Write a program to solve planning problem: The block world problem
 '''  initial
state:
 3 blocks A, B, C and empty holder
A and C are on table, B is on C

 Goal state:
B on table, A on B, C on A, empty holder
'''

blocks = ['A','B','C']
 class state:   def __init__(self, onTable, onA, onB,
onC, holding):
    self.onTable = onTable
self.onA = onA
self.onB = onB
self.onC = onC
self.holding = holding
```

```
   def
displayState(self):
    print("\n*****************State**********************\nholding
: ",s1.holding,"\non Table: ",s1.onTable,"\non A: ",s1.onA,"\non B: ",s
1.onB,"\non C: ",s1.onC)



# initial state
onTable = ['A','C']
onA = "clear" onB =
"clear" onC = "B"
holding = "none"

#goal state
goalOnTable = ['B']
goalOnA = "C" goalOnB
= "A" goalOnC =
"clear" goalHolding =
"none"
 s1 = state(onTable, onA, onB, onC, holding) s2 =
state(goalOnTable, goalOnA, goalOnB, goalOnC,goalHolding)
 def unstack(x):   if
s1.holding=="none":
if s1.onA == x:
s1.onA = "clear"
elif s1.onB == x:
s1.onB = "clear"
elif s1.onC == x:
s1.onC = "clear"
s1.holding = x
s1.displayState()
 def
putDown():
  if s1.holding!="none":
x = s1.holding
s1.onTable.append(x)
s1.holding = "none"
s1.displayState()   else:
    print("No block to put down")
 def pickUp(x):   if
s1.holding=="none":
s1.holding = x
s1.onTable.remove(x)
```

```
    s1.displayState()
else:
    print("Hand is not empty")
 def
on(x,y):
  s1.holding = "none"
return x," is on ",y
 def stack(x,y):
if s1.holding==x:
    s1.holding = "none"
if y=="A":       s1.onA
= x     elif y=="B":
s1.onB = x     elif
y=="C":       s1.onC =
x     s1.displayState()


s1.displayState()

#while s1!=s2: if holding=="none":   if goalOnTable.__contains__('A')
and not onTable.__contains__('A'):
    unstack('A')     putDown()   if goalOnTable.__contains__('B') and
not onTable.__contains__('B'):
    unstack('B')     putDown()   if goalOnTable.__contains__('C') and
not onTable.__contains__('C'):
    unstack('C')
putDown()
   x =
goalOnA   y =
goalOnB   z =
goalOnC
   if x!="clear" and
onTable.__contains__(x):
    pickUp(x)     stack(x,'A')   if
y!="clear" and onTable.__contains__(y):
    pickUp(y)     stack(y,'B')   if
z!="clear" and onTable.__contains__(z):
    pickUp(z)
stack(z,'C')
```

```
print("Done!")
```

Output:

```
********************State************************
holding:  none  on Table:  ['A', 'C']  on A:  clear
on B:  clear  on C:  B

********************State************************
holding:  B  on Table:  ['A', 'C']  on A:  clear
on B:  clear  on C:  clear

********************State************************
holding:  none
on Table:  ['A', 'C', 'B']
on A:  clear  on B:  clear
on C:  clear

********************State************************
holding:  C  on Table:  ['A', 'B']  on A:  clear
on B:  clear  on C:  clear

********************State************************
holding:  none  on Table:  ['A', 'B']  on A:  C
on B:  clear  on C:  clear

********************State************************
holding:  A  on Table:  ['B']  on A:  C  on B:
clear  on C:  clear

********************State************************
holding:  none  on Table:  ['B']  on A:  C  on B:
A  on C:  clear Done!
```