4) Consider S and T as variables and the following relation representing the relationships:

- **(i) a: ¬(SVT)**
- **(ii) b: (S&T)**
- **(iii) c: TV¬T**
- **(iv) d: ¬(S⬄S)**
- **(v) e: ¬S → ¬T**

Analyze the following for PL-TT entailment and show whether

(i). ‘a’ entails ‘b’,

(ii). ‘a’ entails ’c’,

(iii). ‘a’ entails ’d’ and

(iv). ‘a’ entails ’e’

```
N = 4

def main():


    s = [1,0,1,0]

    t = [1,1,0,0]

    a=[]
```

```
b=[]
c=[]
d=[]
e=[]


for i in range(N):
    a.append(not(s[i] or t[i]))
    b.append(bool(s[i] and t[i]))
    c.append(bool(t[i] or(not(t[i]))))
    d.append(not(bidir(s[i],s[i])))
    e.append(imp((not(s[i])),(not(t[i]))))
print("Truth table of a: ",a)
print("Truth table of b: ", b)
print("Truth table of c: ", c)
print("Truth table of d: ", d)
print("Truth table of e: ", e)

p=entails(a, b)
```

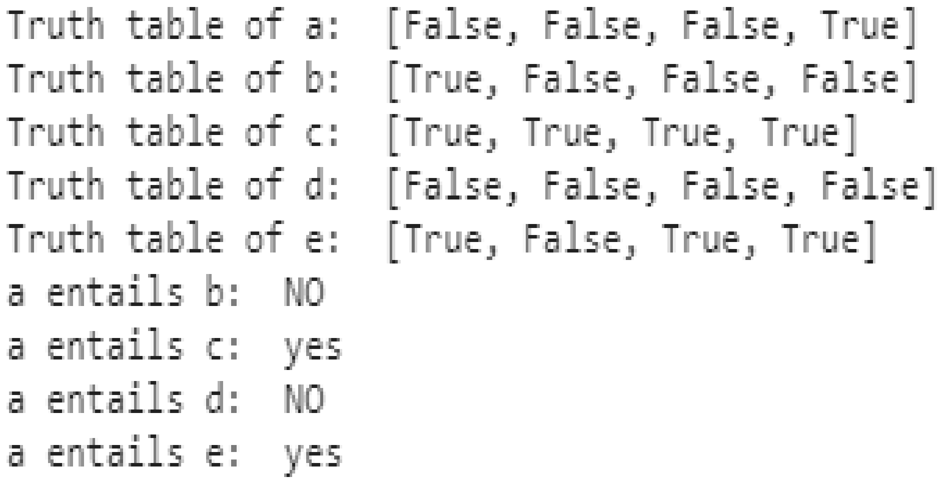
```
    q=entails(a,c)
    r=entails(a, d)
    s=entails(a, e)
    print("a entails b: ",p)
    print("a entails c: ", q)
    print("a entails d: ", r)
    print("a entails e: ", s)

def imp(j,k):
   return (not(j)) or k

def bidir(j,k):
    return (imp(j,k) and imp(j,k))
def entails(m,n):
    #for i in j:
    for i in range(N):
        for j in range(N):
            if (m[i] and n[j]== 1):
                if(i==j):
                    return "yes"
                    break

    return "NO"
if __name__ == '__main__':
    main()
```

OUTPUT



```
Truth table of a:  [False, False, False, True]
Truth table of b:  [True, False, False, False]
Truth table of c:  [True, True, True, True]
Truth table of d:  [False, False, False, False]
Truth table of e:  [True, False, True, True]
a entails b:  NO
a entails c:  yes
a entails d:  NO
a entails e:  yes
```