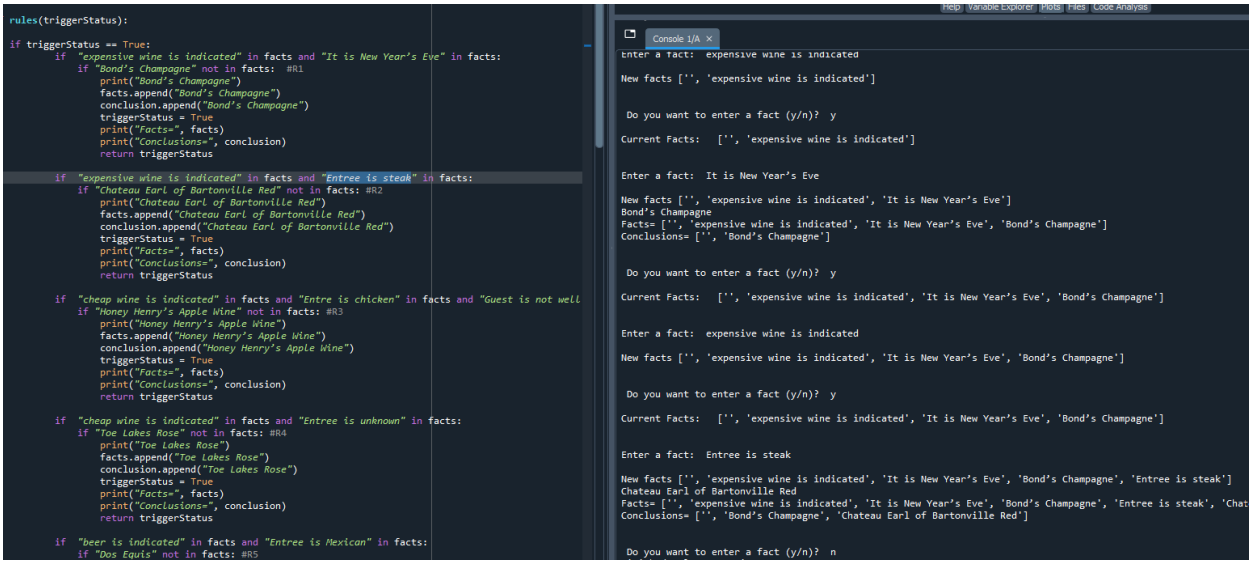
Final Exam ITS 265

Ghadir Alfadhl

Question 1: Rule-based Expert System

Implement in Python a rules function that implements the following bartender rule-base for a forwardchaining expert system. You do not have to write a full application, only implement the rule-base in code. Make sure appropriate conditions and flags are set to support forward-chaining.

R1	If	expensive wine is indicated
		It is New Year's Eve
	Then	Bond's Champagne
R2	If	expensive wine is indicated
		Entrée is steak
	Then	Chateau Earl of Bartonville Red
R3	If	cheap wine is indicated
		Entre is chicken
		Guest is not well liked
	Then	Honey Henry's Apple Wine
R4	If	cheap wine is indicated
		Entrée is unknown
	Then	Toe Lakes Rose
R5	If	beer is indicated
		Entrée is Mexican
	Then	Dos Equis
R6	If	beer is indicated
	Then	Coors
R7	If	guest is a health nut
	Then	Glop
R8	If	guest is a health nut
		Carrots are not to be served
	Then	carrot juice
R9	If	wine is indicated
		Guest should be impressed
	Then	Expensive wine is indicated
R10	If	wine is indicated
	Then	cheap wine is indicated
R11	If	guest is sophisticated
	Then	wine is indicated
R12	If	entrée is Mexican
	Then	beer is indicated
R13	If	guest is not well liked
		Entrée is catered by Not-so-Good Caterers
	Then	beer is indicated
B14	If	guest does not drink alcohol
	Then	water



Code:

```
from typing import List

facts: List[str] = [""]
conclusion: List[str] = [""]
AskQ = True
RuleTriggered = True

def rules(triggerStatus):

    if triggerStatus == True:
        if  "expensive wine is indicated" in facts and "It is New Year's Eve" in facts:
            if "Bond's Champagne" not in facts:  #R1
                print("Bond's Champagne")
                facts.append("Bond's Champagne")
                conclusion.append("Bond's Champagne")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if  "expensive wine is indicated" in facts and "Entree is steak" in facts:
            if "Chateau Earl of Bartonville Red" not in facts: #R2
                print("Chateau Earl of Bartonville Red")
                facts.append("Chateau Earl of Bartonville Red")
                conclusion.append("Chateau Earl of Bartonville Red")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
```

```
            return triggerStatus

        if  "cheap wine is indicated" in facts and "Entre is chicken" in facts and "Guest is not well liked"
in facts:
            if "Honey Henry's Apple Wine" not in facts: #R3
                print("Honey Henry's Apple Wine")
                facts.append("Honey Henry's Apple Wine")
                conclusion.append("Honey Henry's Apple Wine")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if  "cheap wine is indicated" in facts and "Entree is unknown" in facts:
            if "Toe Lakes Rose" not in facts: #R4
                print("Toe Lakes Rose")
                facts.append("Toe Lakes Rose")
                conclusion.append("Toe Lakes Rose")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if  "beer is indicated" in facts and "Entree is Mexican" in facts:
            if "Dos Equis" not in facts: #R5
                print("Dos Equis")
                facts.append("Dos Equis")
                conclusion.append("Dos Equis")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if "beer is indicated" in facts: #R6
            if "Coors" not in facts:
                print("Coors")
                facts.append("Coors")
                conclusion.append("Coors")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if "guest is a health nut" in facts: #R7
            if "Glop" not in facts:
                print("Glop")
                facts.append("Glop")
                conclusion.append("Glop")
                triggerStatus = True
```

```
        print("Facts=", facts)
        print("Conclusions=", conclusion)
        return triggerStatus

    if  "guest is a health nut" in facts and "Carrots are not to be served" in facts:
       if "carrot juice" not in facts: #R8
          print("carrot juice")
          facts.append("carrot juice")
          conclusion.append("carrot juice")
          triggerStatus = True
          print("Facts=", facts)
          print("Conclusions=", conclusion)
          return triggerStatus

    if  "wine is indicated" in facts and "Guest should be impressed" in facts:
       if "Expensive wine is indicated" not in facts: #R9
          print("Expensive wine is indicated")
          facts.append("Expensive wine is indicated")
          conclusion.append("Expensive wine is indicated")
          triggerStatus = True
          print("Facts=", facts)
          print("Conclusions=", conclusion)
          return triggerStatus

    if  "wine is indicated" in facts: #R10
       if "cheap wine is indicated" not in facts:
          print("cheap wine is indicated")
          facts.append("cheap wine is indicated")
          conclusion.append("cheap wine is indicated")
          triggerStatus = True
          print("Facts=", facts)
          print("Conclusions=", conclusion)
          return triggerStatus

    if  "guest is sophisticated" in facts: #R11
       if "wine is indicated" not in facts:
          print("wine is indicated")
          facts.append("wine is indicated")
          conclusion.append("wine is indicated")
          triggerStatus = True
          print("Facts=", facts)
          print("Conclusions=", conclusion)
          return triggerStatus

    if  "entree is Mexican" in facts: #R12
       if "beer is indicated" not in facts:
          print("beer is indicated")
          facts.append("beer is indicated")
          conclusion.append("beer is indicated")
```

```
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if  "guest is not well liked" in facts and "Entree is catered by Not-so-Good Caterers" in facts:
            if "beer is indicated" not in facts: #R13
                print("beer is indicated")
                facts.append("beer is indicated")
                conclusion.append("beer is indicated")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        if  "guest does not drink alcohol" in facts:
            if "water" not in facts: #R14
                print("water")
                facts.append("water")
                conclusion.append("water")
                triggerStatus = True
                print("Facts=", facts)
                print("Conclusions=", conclusion)
                return triggerStatus

        else:
            return triggerStatus
    else:
        triggerStatus = False
        print("Conclusion reached", conclusion)
        return triggerStatus


def GetData(triggerStatus):
    if triggerStatus == True:
        Ans = input("\n Do you want to enter a fact (y/n)?  ")
        if Ans == 'y':
            print("\nCurrent Facts:  ", facts)

            factInput = input("\nEnter a fact:  ")
            #Check whether it is in the list
            if factInput not in facts:
                facts.append(factInput)
                #facts.append("Animal has hair")
            print("\nNew facts", facts)
            return True
        else:
            return False
    else:
```

```
        print("Conclusion reached: ", conclusion)
        return

# Main
while (AskQ):
    AskQ = GetData(RuleTriggered)
    if RuleTriggered == True:
        RuleTriggered = rules(RuleTriggered)


print("Finished Rule Processing")
```