Assignment # 2
Discrete Structure

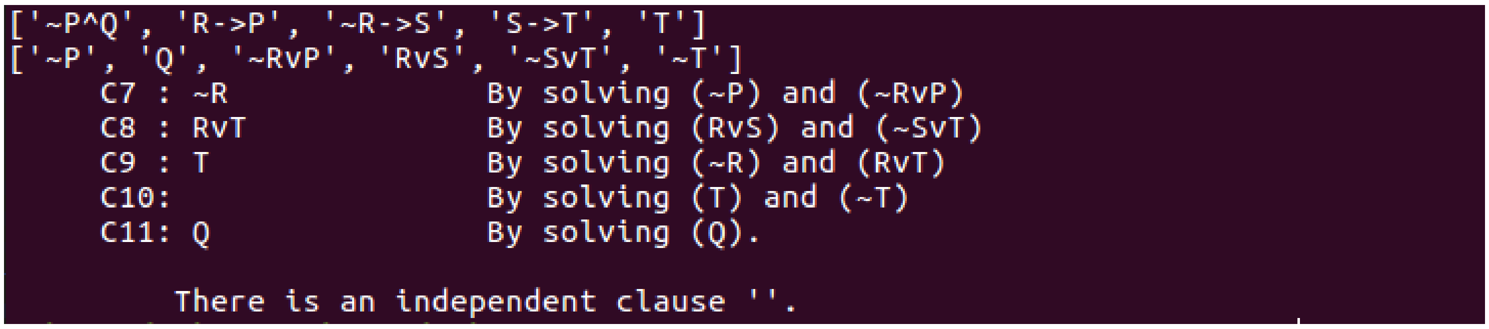
by: Muhammad Ahsan
P17-6142
Section-C
to: Dr. Nauman Azam

Code in Python 3

```
#      EXAMPLE=8
def makeClause(st):
    #print('Use -> for implies \nUse ^ for and \nUse v for OR \nUse ~ for negation\n')
    #str=input("enter a string of premices :")
    clause=[]
    #st="P->Q.~P->R.R->S.~Q->S"
    st=st.split('.')
    size=len(st)-1
    print(st)
    for p in st:
        temp=p.split('->')

        if(len(temp)!=1):
            if(temp[0][0]!='~'):
                temp[0]='~'+temp[0]
                clause.append(temp[0]+'v'+temp[1])
            elif(temp[0][0]=='~'):
                temp[0]=temp[0][1]
                clause.append(temp[0]+'v'+temp[1])
        else:
            temp=p.split('^')
            if(len(temp)==1):
                if(temp[0][0]=='~'):
                    temp[0]=temp[0][1]
                else:
                    temp[0]='~'+temp[0]
                clause.append(temp[0])
            else:
                clause.append(temp[0])
                clause.append(temp[1])

        if(st[size]==p):
            size=len(clause)-1
            temp=clause[size].split('v')
            if(len(temp)!=1):
                if(temp[0][0]=='~'):
                    temp[0]=temp[0][1]
                else:
                    temp[0]='~'+temp[0]
                if(temp[1][0]=='~'):
                    temp[1]=temp[1][1]
                else:
                    temp[1]='~'+temp[1]
                clause[size]=temp[0]
                clause.append(temp[1])
            else:
                if(temp[0][0]!='~'):
                    temp[0]='~'+temp[0][0]
                else:
```

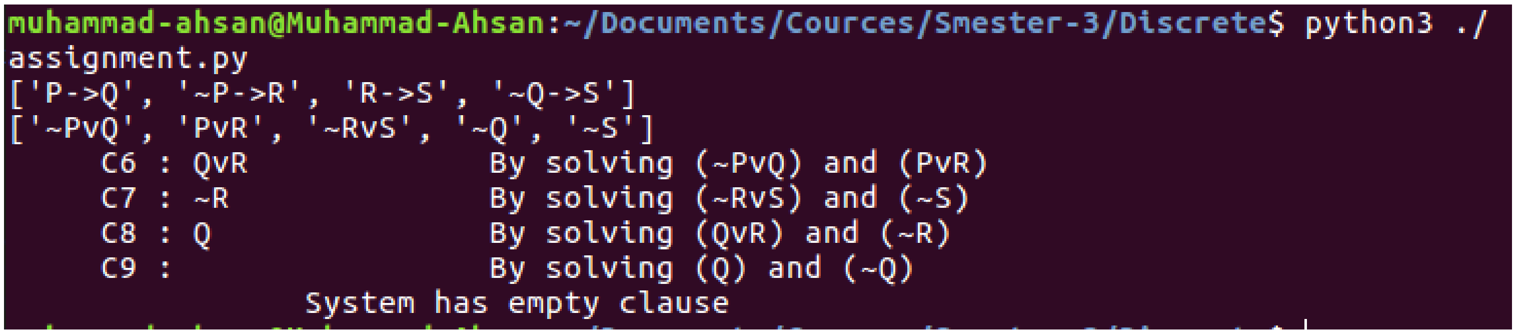
```
                temp[0]=temp[0][1]

    print(clause)
    return clause
def resolution(a, b):
    a = a.split('v')
    b = b.split('v')
    x = ""
    y = ""
    for i in range(len(a)):
        for j in range(len(b)):            # checking principle of resolution
            if (a[i][0] == '~' and b[j][0] == a[i][1]) or (b[j][0] == '~' and b[j][1] == a[i][0]):
                a.pop(i)
                b.pop(j)
                if a == []:
                    x = 'v'.join(b)
                    return x
                if b == []:
                    x = 'v'.join(a)
                    return x
                x = 'v'.join(a)
                y = 'v'.join(b)
                return x + 'v' +y
    return None                         #if not found any premis to apply resolution


def checkClauses():
    # enter premices here
    d="~P^Q.R->P.~R->S.S->T.T"       # premices
    clause=makeClause(d)
    i = 1
    n = len(clause) + 1
    count = 0
    while clause != ['']:               # checking for empty clause
        for i in range(count+1, len(clause)):
            x = resolution(clause[count],clause[i])
            if x != None:
                print( '{:5}C{:2}: {:15} By solving ({}) and ({})'.format("",str(n),x,clause[count],clause[i]))
                clause[count] = x
                clause.pop(i)
                n+=1
                break;
        if len(clause) == 2 and clause[0] == "":
            print( '{:5}C{:2}: {:15} By solving ({}).'.format("",str(n),clause[1],clause[1]))
            print("\n\t There is an independent clause '{}'.".format(clause[0]))
            return 0
        if len(clause) == 2 and clause[1] == "":
            print( '{:5}C{:2}: {:15} By solving ({}).'.format("",str(n),clause[0],clause[0]))
            print("\n\t There is an independent clause '{}'.".format(clause[0]))
            return 0
        if len(clause) == 2 and clause[0] == clause[1]:
            print( '{:5}C{:2}: {:15} By solving ({}) and ({})'.format("",str(n),clause[0],clause[0],clause[1]))
            print("\n\t There is an independent clause '{}'.".format(clause[0]))
            return 0
        if count != len(clause) and len(clause) != 2:
            count+=1
        else:
            count = 0


    return 1

checkClauses()        #   main execution of programm start from here
```

Example#6



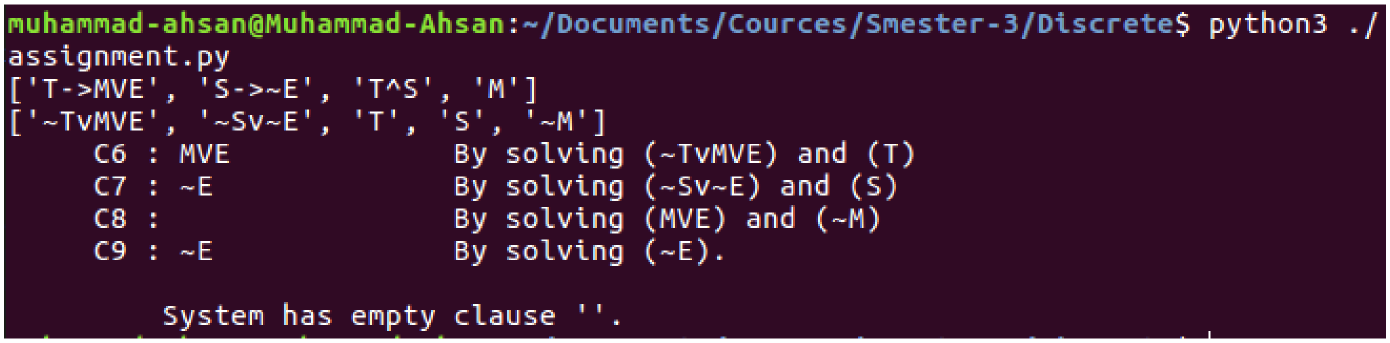
```
['~P^Q', 'R->P', '~R->S', 'S->T', 'T']
['~P', 'Q', '~RvP', 'RvS', '~SvT', '~T']
     C7 : ~R                By solving (~P) and (~RvP)
     C8 : RvT               By solving (RvS) and (~SvT)
     C9 : T                 By solving (~R) and (RvT)
     C10:                   By solving (T) and (~T)
     C11: Q                 By solving (Q).

         There is an independent clause ''.
```

Example#7



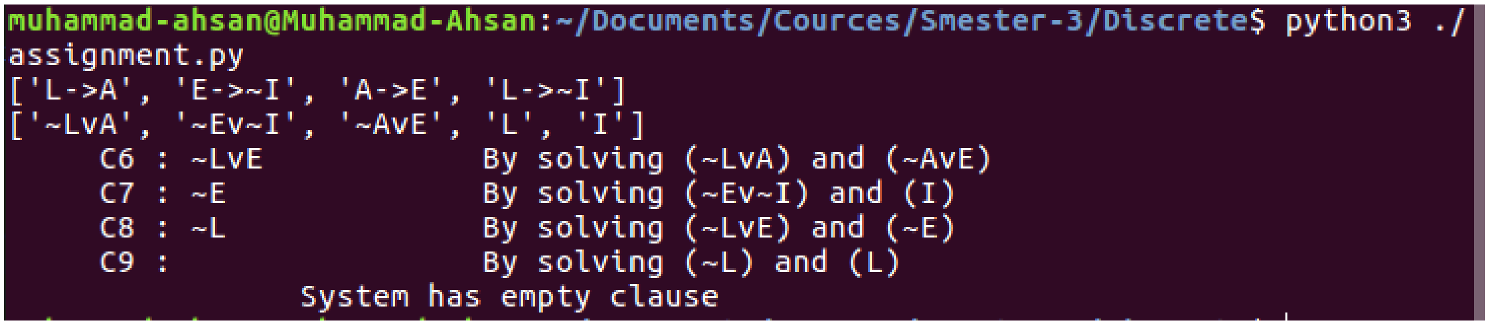
```
muhammad-ahsan@Muhammad-Ahsan:~/Documents/Cources/Smester-3/Discrete$ python3 ./
assignment.py
['P->Q', '~P->R', 'R->S', '~Q->S']
['~PvQ', 'PvR', '~RvS', '~Q', '~S']
     C6 : QvR               By solving (~PvQ) and (PvR)
     C7 : ~R                By solving (~RvS) and (~S)
     C8 : Q                 By solving (QvR) and (~R)
     C9 :                   By solving (Q) and (~Q)
                System has empty clause
```

Example#8



```
muhammad-ahsan@Muhammad-Ahsan:~/Documents/Cources/Smester-3/Discrete$ python3 ./
assignment.py
['T->MVE', 'S->~E', 'T^S', 'M']
['~TvMVE', '~Sv~E', 'T', 'S', '~M']
     C6 : MVE                By solving (~TvMVE) and (T)
     C7 : ~E                 By solving (~Sv~E) and (S)
     C8 :                    By solving (MVE) and (~M)
     C9 : ~E                 By solving (~E).

         System has empty clause ''.
```

Example#9



```
muhammad-ahsan@Muhammad-Ahsan:~/Documents/Cources/Smester-3/Discrete$ python3 ./
assignment.py
['L->A', 'E->~I', 'A->E', 'L->~I']
['~LvA', '~Ev~I', '~AvE', 'L', 'I']
     C6 : ~LvE               By solving (~LvA) and (~AvE)
     C7 : ~E                 By solving (~Ev~I) and (I)
     C8 : ~L                 By solving (~LvE) and (~E)
     C9 :                    By solving (~L) and (L)
                System has empty clause
```