Experiment - 6 Finding first of a grammar

Name: **Rollno :**
Class : TE.CO **Batch :**

#Source code

```
__author__ = 'Shadab Shaikh'
__title__ = 'Finding first of a set from a grammar'
__date__ = '26-02-2019'
__version__ = '1.0'

print('Author          : ' + __author__)
print('Title           : ' + __title__)
print('Date            : ' + __date__)
print('Version         : ' + __version__)

grammararr=[]                   #stores the grammar maintaining the index
fst=[]                          #stores the final result of first set
inputs=""                       #taking input from user
inputm=""                       #continuity of production
s1=""                           #acting as a pointer to compare and find the left most
variable
first=[]
while(inputs!='no'):
        grammar = input("\nEnter the grammar left should be variable following with ->
format eg: S->a\n")
        grammar=grammar.replace(" ","")         #replacing whitespaces with none
        if(grammar[0].islower()):
                grammar[0].upper()              #making left most as variable
        grammararr.append(grammar)              #storing into list
        inputs = input("\nPress no to stop writing productions or write anything to continue")
                                                #asking for the continuity of grammar


def searchprod(grammararr,s1,k,n):
        """function to search production variable if uppercase is found while parsing."""
        for i in range(len(grammararr)):
                if(grammararr[i][0]==s1):       #if leftmost variable matches with the s1 value
                        if(grammararr[i][3].isupper()):
                                                #checking if the 3rd index element is uppercase
                                s1=grammararr[i][3]         #if yes then reassigning s1
                                searchprod(grammararr,s1,k,n)#recursively calling this function
                        elif(grammararr[i][3:].islower()):
                                                #checking if the 3rd and rest index is lowecase
```

```
                    inputm=input("\nTerminal with multiple instances found do
                    you want to seperate  "+grammararr[i][3:]+" terminal for
                    "+grammararr[k][0]+"  press sep  ")
                        #asking if the rest of terminal are combined or seperated
                    if(inputm=='sep'):
                        fst.append("first{"+grammararr[k][0]+"}="
                        +grammararr[i][3])          #considering only the 3rd
                                                    #index element
                    else:
                        fst.append("first{"+grammararr[k][0]+"}="
                        +grammararr[i][3:])
                            #considering all the remaining index element
                else:
                    if(grammararr[i][3]=='#'):
                                #checking to see whether all the
                                #production have became epsilon
                        if(grammararr[k][3+n] is not None):
                        #if index element is not none of the parsing variable
                            if(grammararr[k][3+n].islower()):
                            #checking if the next element is lowercase
                                fst.append("first{"+grammararr[k][0]
                                +"}="+grammararr[k][3+n])
                                #assigning it into first of a result set
                            else:
                                s1=grammararr[k][3+n]
                                #updating s1 value
                                fst.append("first{"+grammararr[k][0]+
                                "}="+grammararr[i][3])      #assigning
                                #epsilon to each of the result set
                        else:
                            fst.append("first{"+grammararr[k][0]+
                            "}="+grammararr[i][3])
                            #if the next parsing index is none then assigning
                            #epsilon
                    else:
                            fst.append("first{"+grammararr[k][0]+"
                            }="+grammararr[i][3])
                            #by default making it as epsilon


def findfirst(grammararr,k):
    """function to search production variable if uppercase is found while parsing."""
    if(grammararr[k][3].isupper()):             #checking if the 3rd element of a list
                                                #index position is uppercase(variable)
        s1=grammararr[k][3]                     #assigning it to s1
        searchprod(grammararr,s1,k,1)           #calling the searchprod function
```

```
        elif(grammararr[k][3:].islower()):      #if the 3rd element and all are lowercase
                inputm=input("\nTerminal with multiple instances found do you want to
                seperate "+grammararr[k][3:]+" terminal for "+grammararr[k][0]+" press
                sep ")
                #asking if the rest of terminal are combined or seperated
                if(inputm=='sep'):
                        fst.append("first{"+grammararr[k][0]+"}="+grammararr[k][3])
                                        #considering only the 3rd index element
                else:
                        fst.append("first{"+grammararr[k][0]+"}="+grammararr[k][3:])
                                        #considering all the remaining index element
        else:
                fst.append("first{"+grammararr[k][0]+"}="+grammararr[k][3])
                                                #by default making it as 3rd element
        k+=1                                    #incrementing k by 1
        if(k<len(grammararr)):                  #until k is less than grammar list
                findfirst(grammararr,k)         #recursively calling findfirst funtion

findfirst(grammararr,0)                 #calling findfirst function

fst=list(set(fst))                      #getting all the unique results
fst.sort()                              #sorting the list
print(*fst, sep = "\n")                 #printing the final result
```

#**Output**

Sample Grammar inputs

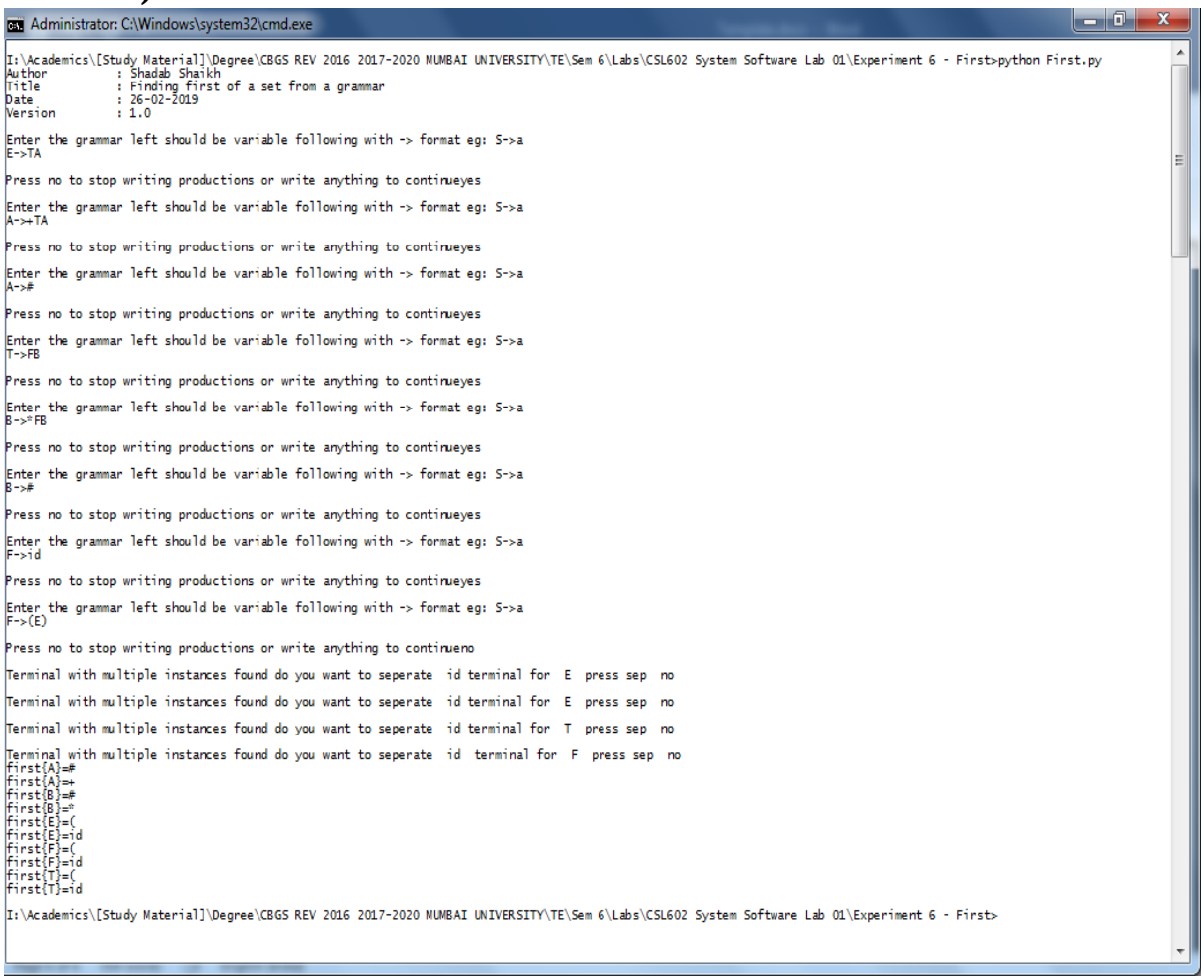
1. E->TA
   A->+TA/€
   T->FB
   B->*FB/€
   F->id/(E) #(Case of non-splitting of terminal)

2. S->(S)/ €

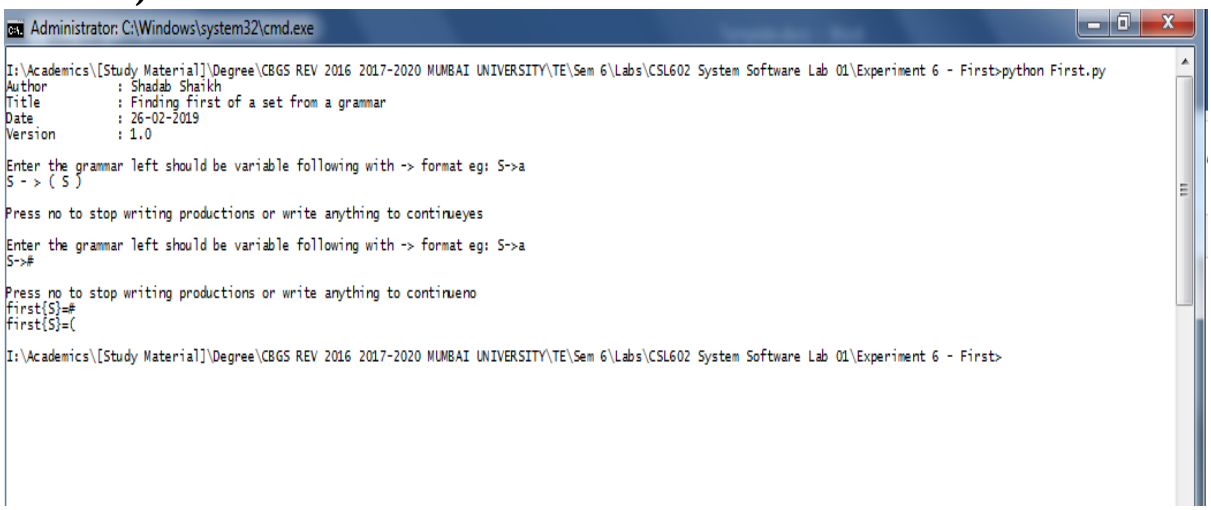
3. S->aABb
   A->C/€
   B->d/€

4. S->ACB/CbB/Ba
   A->da/BC
   B->g/€
   C->b/€
   (Case of € replacement and splitting terminal)

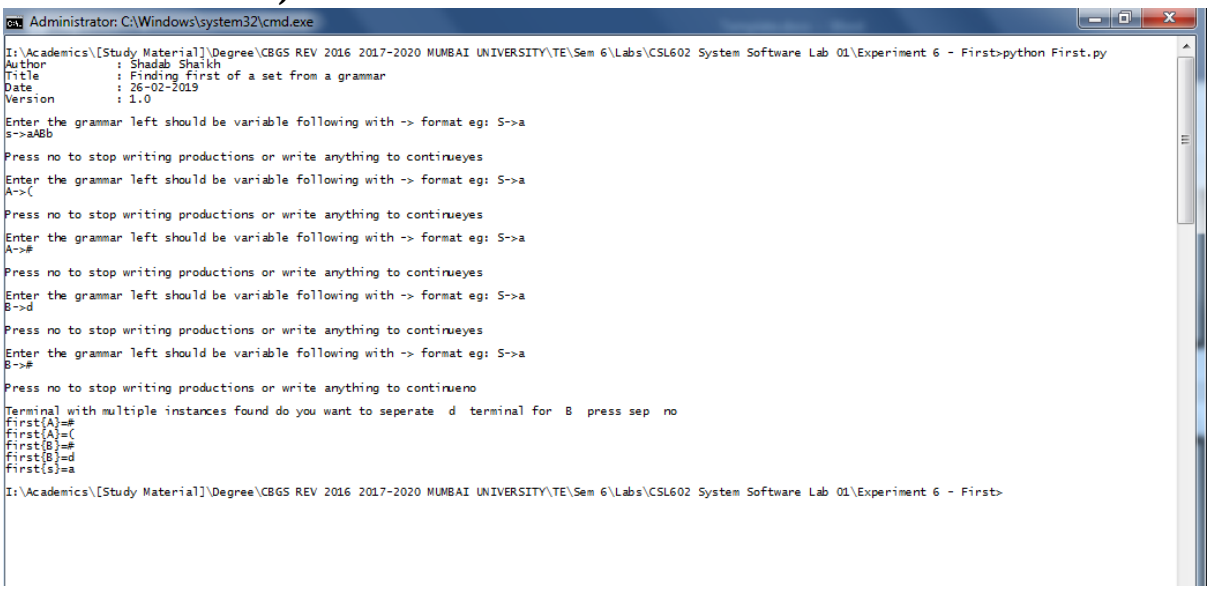
1. **Output for the first sample grammar (Case of non-splitting of terminal)**



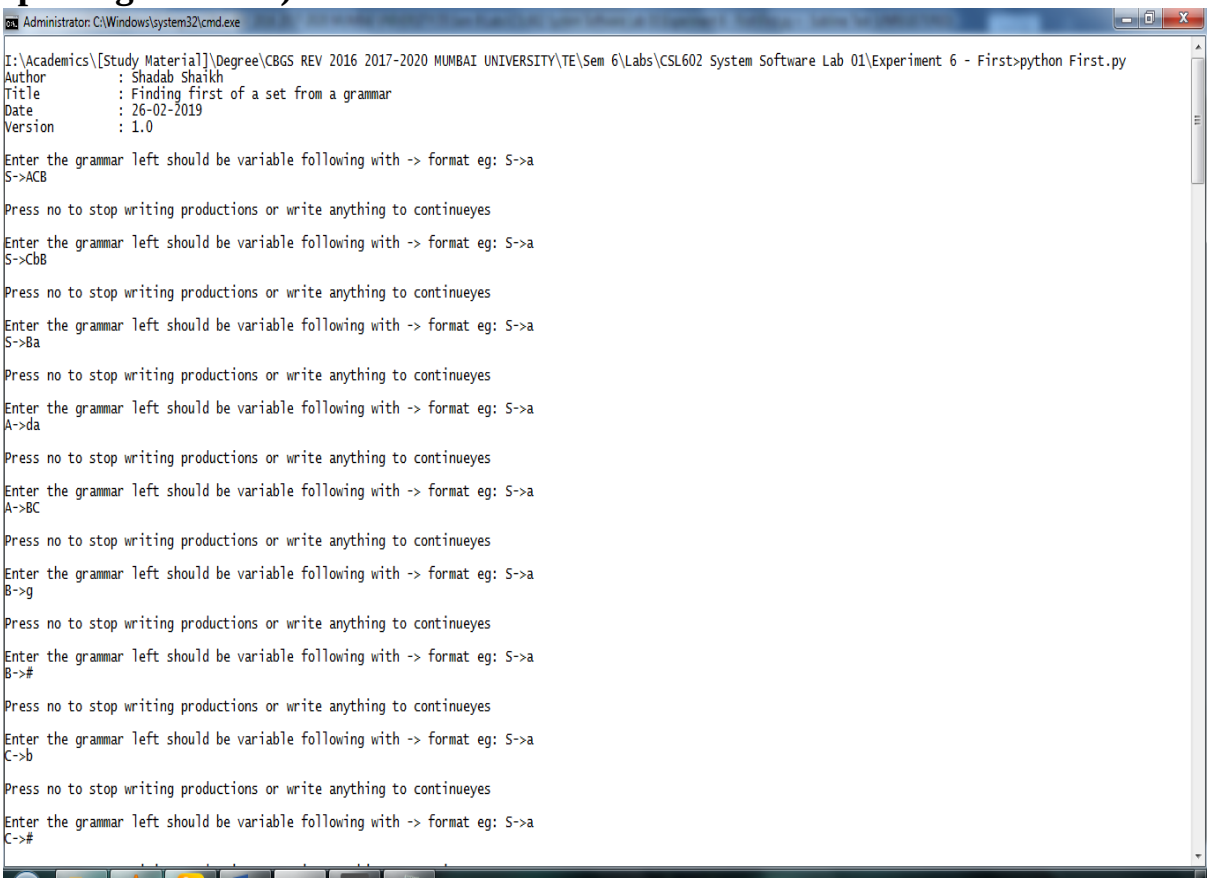
2. **Output for the second sample grammar (automatic whitespace removal)**



3. **Output for the third sample grammar (automatic uppercase for the left-most element)**



4. **Output for the fourth sample grammar (Case of € replacement and splitting terminal)**



```
Terminal with multiple instances found do you want to seperate  da terminal for  S  press sep  sep

Terminal with multiple instances found do you want to seperate  g terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  b terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  g terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  b terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  b terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  g terminal for  S  press sep  no

Terminal with multiple instances found do you want to seperate  da  terminal for  A  press sep  sep

Terminal with multiple instances found do you want to seperate  g terminal for  A  press sep  no

Terminal with multiple instances found do you want to seperate  b terminal for  A  press sep  no

Terminal with multiple instances found do you want to seperate  g  terminal for  B  press sep  no

Terminal with multiple instances found do you want to seperate  b  terminal for  C  press sep  no
first{A}=#
first{A}=b
first{A}=d
first{A}=g
first{B}=#
first{B}=g
first{C}=#
first{C}=b
first{S}=#
first{S}=a
first{S}=b
first{S}=d
first{S}=g

I:\Academics\[Study Material]\Degree\CBGS REV 2016 2017-2020 MUMBAI UNIVERSITY\TE\Sem 6\Labs\CSL602 System Software Lab 01\Experiment 6 - First>
```