Examen "Beginselen van Programmeren", Deel 2 (28 januari 2019)

Toegangscontrole

Een bedrijf wil per lokaal bijhouden welke personen tot die lokalen toegang hebben, om daarna een aantal gegevens hieruit te kunnen opvragen. Zo'n informatie wordt typisch bijgehouden in een zgn. 'Access Control List', of ACL.

Bedenk een gepaste data-structuur die deze informatie in een Python-programma kan bijhouden, en implementeer dan de functies die hieronder zijn opgelijst.

Bemerk:

- Je mag ervan uitgaan dat elk lokaal een unieke naam heeft, en ook de naam van elke persoon uniek is.
- Je hoeft binnen de functies geen validatie van de parameters te doen. Je mag er steeds vanuit gaat dat de parameter correcte en aanvaardbare waarden hebben.
- LET OP! Lees aandachtig de beschrijving van de functies! Als een verzameling (set) als return-waarde gevraagd wordt, is een lijst niet goed!
- Zorg dat de .py-file die je indient geen compilatie-fouten bevat!

Gevraagde functies

```
########################################################################
# Twee functies voor het aanmaken en invullen van de data-structuur
########################################################################

def creeer_ACL ():
    ...
    # deze functie creeert een lege data-structuur, d.w.z. een initiële
    # data-structuur, zonder gegevens over lokalen en personen

def voeg_toegang_toe (acl, lokaal, naam_persoon):
    ...
    # deze functie voegt aan de data-structuur acl' de informatie toe dat de
    # persoon tot het lokaal toegang krijgt; deze functie geeft als return-
    # waarde de aangepaste ACL terug; als een persoon met dezelfde naam eerder
    # al werd toegevoegd aan dit lokaal, dan mag deze persoon geen 2e maal
    # worden toegevoegd.

########################################################################
# Twee functies voor eenvoudige vragen aan de data-structuur
########################################################################

def tot_welke_lokalen_heeft_persoon_toegang (acl, naam_persoon):
    ...
    # geeft een verzameling terug van lokalen waartoe
    # de persoon met die naam toegang heeft; het is mogelijk
    # dat aan deze persoon eerder nog geen toegang aan een
    # lokaal werd verleend; in dat geval geeft deze functie
    # een lege verzameling terug.

def geef_alle_personen_met_toegang (acl):
    ...
    # geeft een verzameling terug van alle personen aan wie eerder
    # toegang tot eender welk lokaal werd verleend

########################################################################
# Enkele meer geavanceerde functies die de data-structuur ondervragen
########################################################################

def wie_heeft_meeste_toegang (acl):
    ...
    # deze functie geeft als return-waarde terug wie toegang heeft tot het grootste
    # aantal lokalen; indien meerdere personen tot evenveel lokalen toegang hebben,
    # dan volstaat het om 1 naam terug te geven; de functie geeft een lege string
    # als resultaat als er nog geen toegangen in de ACL werden opgenomen.
```

```
def geef_aantal_lokalen_waar_deze_personen_samen_binnen_kunnen (acl, personen)
    # de parameter personen bevat een verzameling van personen; de functie
    # geeft als return-value een natuurlijk getal dat aangeeft tot hoeveel
    # lokalen deze groep toegang heeft; de groep heeft toegang tot een lokaal
    # als minstens 1 persoon van de groep toegang tot het lokaal heeft

def geef_alle_mogelijke_groepen(personen):
    ...
    # implementeer deze functie als een RECURSIEVE functie; deze
    # functie genereert een lijst van alle mogelijke (deel)groepen
    # van de personen in de verzameling die als parameter werd meegegeven;
    # elke groep wordt voorgesteld als een verzameling;

def geef_kleinste_groep_die_samen_overal_binnen_kunnen (acl):
    ...
    # deze functie gaat op zoek naar een groep van 1 of meer personen
    # die, als ze allemaal samenwerken, toegang heeft tot alle
    # lokalen waarvoor informatie werd ingegeven; meer bepaald
    # moet deze functie een groep teruggeven met een zo klein
    # mogelijk aantal personen; als er meerdere zo'n groepen zijn,
    # dat volstaat het om één zo'n groep terug te geven; de return-value
    # van de functie is een verzameling van personen in die groep.

##############################################################################
# Hieronder vind je 1 voorbeeld van een hoofdprogramma. Dit pas je uiteraard
# aan om je programma te testen, en wordt op zich niet verbeterd.
##############################################################################

def main ():

    toegangslijst = creeer_ACL()

    voeg_toegang_toe(toegangslijst, "vergaderlokaal A", "Tobias")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal B", "Tobias")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal C", "Tobias")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal A", "Maria")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal A", "Maria")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal D", "Tobias")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal A", "Julie")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal A", "Kasper")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal B", "Kasper")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal C", "Julie")
    voeg_toegang_toe(toegangslijst, "vergaderlokaal D", "Kasper")

    julies_lokalen = tot_welke_lokalen_heeft_persoon_toegang(toegangslijst, "Julie")
        # geeft een set terug met 2 lokalen, nl. vergaderlokalen A en C
        # en kent toe aan de variabele julies_lokalen
    alle_personen = geef_alle_personen_met_toegang (toegangslijst)
        # geeft een verzameling terug met de namen Tobias, Maria, Julie, Kasper
        # en kent toe aan de variabele alle_personen
    meest_bevoegde = wie_heeft_meeste_toegang (toegangslijst)
        # geeft 1 naam terug, nl. Tobias
        # en kent toe aan de variabele meest_bevoegde
    deelgroepen = geef_alle_mogelijke_groepen( { "Tobias", "Julie", "Kasper"} )
        # geeft als resultaat een lijst met de volgende deelgroepen
        #  [  set()  , {"Tobias"}, {"Julie"}, {"Kasper"}, {"Tobias","Julie"},
        #     {"Tobias","Kasper"}, {"Julie","Kasper"}, {"Tobias","Julie","Kasper"} ]
    kleinste_groep = geef_kleinste_groep_die_samen_overal_binnen_kunnen (toegangslijst)
        # 2 mogelijkheden: ofwel een verzameling met daarin Tobias en Kasper
        #                  ofwel een verzameling met daarin Julie en Kasper
        # één van beide wordt toegekend aan de variabele kleinste_groep

##############################################################################
# Gebruik zeker onderstaande lijnen om je eigen hoofdprogramma op te
# starten.
##############################################################################

if __name__ == "__main__":
    main()
```