a)

1) Veronika - Luka
   Petra - Anže
   Barbara - Tomaž
   Katarina - Jan
2) Veronika - Anže
   Barbara - Tomaž
   Katarina -Jan
   Petra - Luka

b)

i)

Graf v prilogi

ii)

```
import numpy as np
import math

preference_moski = {
    "Luka": ["Veronika", "Petra", "Barbara", "Katarina"],
    "Anže": ["Petra", "Veronika"],
    "Tomaž": ["Barbara"],
    "Jan": ["Veronika", "Barbara", "Katarina"]
}
preference_zenske = {
    "Veronika": ["Luka", "Anže", "Tomaž"],
    "Petra": ["Anže", "Luka"],
    "Barbara": ["Tomaž", "Jan"],
    "Katarina": ["Tomaž", "Anže"]
}


def main():
   for i in range(0, math.factorial(len(preference_moski))):  # probamo najveckrat n!

       np.random.permutation(preference_moski)  # shuffle preferences
       np.random.permutation(preference_zenske)  # shuffle preferences

       engagements = {}
       for zenska, preference in preference_zenske.items():
           for preferenca in preference:
               if zenska in preference_moski[preferenca]:
                   # obstaja trdno povezana povezava
                   engagements[zenska] = preference

       ##do tu imamo samo mocno povezane komponente, kjer ima zenska več preferenc ki ji
pašejo in
       # jo imajo fantje nazaj radi
```

```
        ##preveris ce je se kaksna zenska brez para
        for zenska, preference in preference_zenske.items():
            if zenska not in engagements.keys():
                for moski, preference in preference_moski.items():
                    if moski not in engagements.values():

                        # zdaj ju imamo ko še nista poročena
                        # zdaj preverimo če ima vsaj en rad drugega
                        if zenska in preference_moski[moski]:
                            engagements[moski] = zenska
                            # in ju združimo


main()
```

iii)

Smiselne so samo dvosmerne povezave, saj se samo tako oba hočeta poročiti en z drugim

Trdni zakon je tisti, kjer nekatera oseba nima več dvosmernih povezav.

c)

Problem se imenuje stable marriage problem

Da je problem sigurno rešljiv, morajo biti polja preferenc napolnjena, saj se lahko zgodi, da ko fant zaprosi dekleta, so vsa ta dekleta že z drugim fantom, potem pa fantu zmanka preferenc
in ostane sam, prav tako pa ostane samo eno dekle.

Isti problem ima lahko samo eno rešitev, če je podana samo ena preferenca.
Če pa se preference ljudi spremenijo, pa algoritem privede do druge rešitve. Algoritem za osebe najde samo najboljše reštive, zato je rešitev samo ena.

Če so polja preferenc napolnjena, so na koncu iteracij vsi poročeni/zaročeni, ker vsak moški zaprosi neko
žensko ob enem trenutku.

Prav tako so stabilne poroke/zaroke, ker če katera poroka ni stabilna, to pomeni da ženska fanta ni
zavrnila za njej boljšo ponudbo.

Ponesreči sem napisal stable marriage problem za b nalogo:

```
# Algoritem sestavljen po: Stable Marriage Problem - Numberphile - Gale-Shapley algorithm
from collections import defaultdict
import numpy as np
import math

def propose(woman, proposed, preference_zenske,rejected, engaged ):
        womans_preferences = preference_zenske[woman]
        next_propose = womans_preferences[0]
```

```
        #prvo pogleda zenska kateri po listi jo je že zavrnil
        if woman in rejected.keys():
                for i in range(len(womans_preferences)):
                        if womans_preferences[i] == rejected[woman]:
                                if(i < len(womans_preferences)-1):
                                        next_propose = womans_preferences[i+1] #izbere
naslednjega
        #ce se ni zarocena ga prosi za roko
        #if(next_propose not in engaged):
        proposed[next_propose].append(woman)

def reject(man, preference_moski, proposed, engaged, rejected):
        mans_preferences = preference_moski[man]
        mens_proposals = proposed[man]

        for preference in mans_preferences:
                #izbere najboljso njemu
                for propose in mens_proposals:# ce ni forever -alone guy aka ce ga je
zaprosila ksna
                        if preference == propose:
                                proposed[man] = [preference] #samo njo izbere, ostale zavrne
                                engaged[man] = preference # se zarocita zacasno

                                #reject women:
                                mens_proposals.remove(preference)

        #zavrnemo
        for failed_proposal in mens_proposals:
                rejected[failed_proposal] = man #pove kdo jo je rejectov

def main():
        ST_PAROV = 4
        engaged = {}

        proposed = defaultdict(list)
        rejected = {}

        preference_moski = {
                "Luka": ["Veronika", "Petra", "Barbara", "Katarina"],
                "Anže": ["Petra", "Veronika"],
                "Tomaž": ["Barbara"],
                "Jan": ["Veronika", "Barbara", "Katarina"]
        }
        preference_zenske = {
                "Veronika": ["Luka", "Anže", "Tomaž"],
                "Petra": ["Anže", "Luka"],
                "Barbara": ["Tomaž", "Jan"],
                "Katarina": ["Tomaž", "Anže"]
        }
        for i in range(math.factorial(ST_PAROV)): ### da ponovimo večkrat - za vse možne
pare...

                while(len(engaged) < ST_PAROV): # ko so vsi engaged ke konec
```

```
                for woman in preference_zenske.keys():
                        propose(woman, proposed, preference_zenske,rejected, engaged)
                for man in preference_moski.keys():
                        reject(man, preference_moski, proposed, engaged, rejected)

        for man, woman in engaged.items():
                print(man, "<3", woman)
        print()

        ##permute --- tako poiščemo več različnih zaročič
        for preferences in preference_zenske.values():
                np.random.permutation(preferences)
        for preferences in preference_moski.values():
                np.random.permutation(preferences)
main()
```