Praktiline ülesanne:

Seiklusmäng – tekstipõhine seiklusmäng, milles mängija peab piiratud arvu käikude jooksul jõudma aardeni.

1. Näidake tervitussõnumit, selgitage mängu eesmärki ja teatage lubatud käikude arvu (7 käiku).
2. Seadistage mängukeskkond - initsialiseerige mängumuutujad: määrake *käiku_jäänud* väärtuseks 7, *aare_leitud* väärtuseks False ja *etapp* väärtuseks 1.
3. Kasuta while-tsüklit, mis töötab seni, kuni *käiku_jäänud* on suurem kui 0 ja *aare_leitud* on False.
4. Näita mängijale tsükli sees enne iga käigu algust järelejäänud käikude arvu.
5. Rakendage etappe ja valikuid.
    - 1. etapp: mängija alustab metsast ja peab valima suuna (põhja, lõuna, ida või lääne). Õige suund on ida.
    - 2. etapp: mängija jõuab jõe äärde ja peab valima ujumise, parve ehitamise või silla leidmise vahel. Õige valik on parv või sild.
    - 3. etapp: mängija ületab jõe ja kohtab teelahkmel. Ta peab valima, kas liikuda vasakule või paremale. Õige valik on vasakule.
    - 4. etapp: Mängija leiab end ukse ees, kus on peidetud aare. Ukse avamiseks peab koodi ära arvama, sellele antakse 5 katset. Kood on juhuslik arv vahemikus 1 kuni 10.
6. Kasutage tsükli sees iga etapi puhul if-elif lauseid, et kontrollida, millise etapiga on tegu (1, 2 või 3), ning esitage mängijale stsenaarium ja valik olenevalt etapist. Mängu alguses panime juba algväärtuse, muutuja *etapp* =1.
7. Kasutage input(), et saada mängija valik ja uuendada *käiku_jäänud*, vähendades seda 1 võrra pärast iga valikut.
8. Kasutage kasutaja sisestuse kontrollimiseks sisseehitatud if-else avaldusi: kui see on õige, siis uuendage etapimuutujat, et minna järgmisesse etappi; kui see on vale, siis kuvage veateade ja laske mängijal uuesti proovida.

   Vihje: kasutage kasutaja sisendi kontrollimisel lower() või upper() sõne meetodeid.
9. Loo salakoodi äraarvamise 4. samm. 4. sammu elif-avalduses genereeri salakoodiks juhuslik täisarv vahemikus 1-10 (importige selleks random mooduli) (loo muutuja *salakood* ja anna sellele väärtus) ning anna mängijale 5 katset selle äraarvamiseks (loo muutuja *katset_jäänud* ja anna sellele väärtus 5).

10. Kasuta sisseehitatud while-tsüklit katsete käsitlemiseks: kui *katset_jäänud* on suurem kui 0, mis tähendab, et mängijal on veel katseid, hankige mängija sisend funktsiooniga input() ja võrrelge seda salakoodiga.
11. Kui mängija arvab koodi õigesti ära, määra *aare_leitud* True, näita teade, et uks on avatud ja aare on leitud, ning katkesta while-tsükkel break'iga.
12. Kui mängija oletus on vale, vähenda katsete arvu *katset_jäänud* ja kuva teade ülejäänud katsetega.
13. Väljaspool mängutsüklit lisage else-lause (while-tsükli sees), mis käivitatakse, kui tsükkel lõpeb katkestusteta (kasutage break). See tähendab, et mängijal on käigud otsa saanud.
14. Pärast mängutsüklit kasutage lauset if-elif-else, et kuvada sobiv teade sõltuvalt sellest, kas mängija võitis, kaotas või tegi liiga palju käike. Kui aare_leitud = True, on mängija võitnud. Kui *käiku_jäänud* <=0 – mängija on teinud liiga palju käike, on mäng läbi. Kui mängija ei arvanud ära salakoodi, mängija kaotas.
15. Testige mängu, veendumaks, et see töötab korralikult ja kas mängija suudab võita, kaotada või teha liiga palju pöördeid.