Name: Advanced RSync	**ID:1**	**Difficulty: A**	**Propose: GDT**
Sunt primite două locaţii care trebuie ţinute sincronizat. O locaţie are următoarea formă: <TIP_LOCATIE>:<Path_in_locatie> Exemple: 1) ftp:user:parola@URL/a.b.c 2) zip:C:\abc\d.zip 3) folder:C:\aaa Scriptul rulează încontinuu şi tine sincronizate cele două locaţii. Mai exact: - Dacă se creaza un fişier într-o locaţie, se duplică fişierul şi în cealaltă locaţie - Dacă se şterge un fişier dintr-o locaţie, se şterge şi din cealaltă locaţie - Dacă se modifică un fişier dintr-o locaţie, se copiază modificarea şi în cealaltă locaţie. La pornirea iniţială, sincronizarea se face în felul următor: - Dacă un fişier există doar pe o locaţie - se copiază şi în cealaltă locaţie - Dacă acelaşi fişier există în ambele locaţii, dar există diferenţe, se copiază cel mai nou fişier (dpdv al timpului în care a fost modificat) de pe una din locaţii pe cealaltă.			
INPUT: advanced_rsync.py <location_1> <location_2> *Ex: advanced_rsync.py ftp:my_user:1234@127.0.0.1/folder zip:C:\aaa*			
OUTPUT: Orice formă de logging este OK (validarea se va face între cele două locaţii).			

Name: Twitter	**ID:2**	**Difficulty: A**	**Propose: CGI**
Folosind API-ul Twitter să se ia cele mai recente x tweet-uri ce contin un anumit hashtag si sa se ploteze pe o harta locaţiile din care au fost postate. Se accepta folosirea oricărei biblioteci atat pentru comunicarea cu Tweeter cat şi pentru vizualizarea datelor.			
INPUT: - Un hashtag			
OUTPUT: Harta in care vor fi marcate locatiile corespunzatoare celor mai recente x tweet-uri ce contin acel hashtag.			

Name: Messenger	**ID:3**	**Difficulty: A**	**Propose: MLA**
Sa se scrie o aplicatie grafica de tipul client-server prin care 2 clienti sa poata comunica intre ei. Fiecare client va avea un id atribuit. Mesajele vor fi trimise unui anumit id si vor fi sub forma de mesaje text. Mesajele vor fi salvate intr-un fisier si se va putea urmari istoricul mesajelor primite/trimise unui anumit client. Id-urile clientilor trebuie sa fie unice. Comunicarea trebuie sa fie criptata, si sa permita si emojy precum si trimitere de poze.			
INPUT: id-ul clientului curent			

OUTPUT: fisierele de logging cu istoricul mesajelor.

Name: Meeting scheduler	**ID:4**	**Difficulty:A**	**Propose: CGI**
Construiti o aplicatie grafica care va putea gestiona meeting-urile (sedintele) cu ajutorul unei baze de date PostgreSQL . Aceasta va permite interactiunea cu utilizatorul printr-un meniu cu urmatoarele comenzi: - Adugarea unei persoane in baza de date - Stabilirea unui viitor meeting (data de inceput, data de sfarsit, lista persoanelor care vor participa) - Afisarea sedintelor dintr-un anumit interval de timp - Export/Import intr-un format de calendar standard (fie ical, fie ics, etc). Se va face export-ul/importul doar catre un singur format. Aplicatia va face validarea input-ului dat de utilizator si va face error handling in sensul ca va afisa un mesaj sugestiv in cazul aparitiei unei exceptii.			
INPUT: Interfata cu meniu interactiv Ex. adauga persoana Ion Popescu adauga sedinta care va incepe la 2020-11-20 14:00, se va termina la 2020-11-20 14:30 si la care vor participa Ion Popescu, Ana Maria afiseaza sedintele din intervalul 2020-11-20 08:00, 2020-11-20 23:59			
OUTPUT: Orice metoda de afisare este OK			

Name: Minesweeper	**ID:5**	**Difficulty: A**	**Propose: RMC**
Se va crea o aplicaţie grafica care va simula jocul Minesweeper. https://www.instructables.com/How-to-play-minesweeper/ De asemenea va exista şi posibilitatea ca tabla de joc sa aibă dimensiuni variabile, să se poată seta numărul de bombe utilizate şi un timp limita (în secunde) în care sa se rezolve tabla.			
INPUT: minesweeper.py			
OUTPUT: interfata grafica cu jocul			

Name: Diff Update	**ID:6**	**Difficulty: A**	**Propose: GDT**
Realizati o aplicaţie care sa producă un update prin diferente intre doua fişiere binare. Aplicaţia primeşte la intrare versiunea curentă a unui fisier binar, precum si mai multe versiune precedente ale acestuia si creaza o lista de comenzi (de tipul insert / delete /			

change) prin care se poate ajunge de la unul din fişierele precedente la fişierul curent. Lista în cauză este encodata într-un fişier binar. Cu acel fişier binar, pornind de la unul dintre fişierele cu versiune mai veche, se poate obţine fişierul cel mai nou.
INPUT: difupdate.py create abc.latest abc.ver1 abc.ver2 abc.ver3 (in cazul de mai sus, se presupune ca ultima versiune este 4) Rezultatul e un fişier cu numele “abc.diff” care conţine informaţiile necesare ca sa se poate ajunge de la versiunea 1 , 2 respectiv 3 la versiunea 4 (latest) **INPUT**: difupdate.py update abc.ver2 abc.diff Rezultatul este ca se va crea un fişier abc.latest obţinut aplicand operaţii de tipul (insert, delete sau change bytes) pe fişierul abc.ver2. Aceste operaţii sunt descrise în abc.diff (ca fiind cele necesare ca sa trecem de la versiunea 2 la versiunea 4 (latest).
OUTPUT:

Name: Custom (un)packer	**ID:7**	**Difficulty: A**	**Propose: NIP**
Creati un tool de impachetare/despachetare fisiere. Tool-ul manipuleaza arhive create dupa un format definit de dezvoltator. Un set minim de comenzi pe care va trebui sa le stie tool-ul sunt: creare_arhiva (cu param 1 fisier, 1 director sau o lista de fisiere) - creaza o arhiva Listare_continut - va lista fisierele din interiorul arhivei si size-ul lor Full_unpack (cu parametru un folder destinatie) - dezarhiveaza toata arhiva Unpack (lista de fisiere in folder de output) - dezarhiveaza doar fisierele respective !!! nu se vor folosi biblioteci python 3rd party si nu aveti voie sa folositi zipFile (nu este nevoie de compresie)			
INPUT: Comenzile listare_continut, full_unpack, unpack			
OUTPUT: Arhiva si fisierele rezultate in urma comenzilor rulate Logurile comenzilor executate precum si erorile aparute			

Name: BingeWatch	**ID:8**	**Difficulty: A**	**Propose: NIP**
Creaţi un tool care monitorizeaza serialele favorite. Într-o bază de date se va păstra numele serialului, link către IMDB, ultimul episod vizionat, data ultimei vizionari, un scor (setat de user pentru serial). Cand va fi rulat tool-ul va lista ce seriale noi apărute nu au fost vizionate în funcţie de scorul serialului. Tool-ul va cauta si traler-uri pe youtube sau upload-uri care au legatura cu un anumit episod dintr-un serial şi va oferi o lista a acestora (respectiv notificari dacă apar altele).			
INPUT: Adaugare serial (link imdb si scor)			

Sergere serial
Modificare scor
snooze/unsnooze (dacă e snoozed un serial nu va apărea în lista de seriale cu episoade noi)
Listare - va lista toate episoadele noi ale serialelor din db (mai puţin cele snoozed)

OUTPUT:
Rezultatul comenzilor împreuna cu logarea activităţilor şi erorile apărute

Name: SongStorage	**ID:9**	**Difficulty: A**	**Propose: NIP**

Intr-un folder (demunit Storage) se vor stoca melodii (mp3,wav, samd). Dezvoltati tool-ul SongStorage care va adauga melodii in Storage. Pe langa fisierele cu melodiile, tool-ul va putea memora intr-o baza de date anumite informatii despre o melodie: nume fisier, artist, nume melodie, data aparitiei precum si o lista de tag-uri.
Tool-ul va permite mai multe operatii, printre care:

- Adaugare melodie (va adauga fisierul in storage si metadate in baza de date)
- Stergere melodie (va sterge atat fisierul cat si metadatele din baza de date)
- Modificare metadate (va permite modificarea metadatelor pentru o anumita melodie)
- Creare “Savelist” - creaza o arhiva cu melodiile selectate dupa anumite criterii
- Search - cauta o melodii dupa anumite criterii si intoarce metadatel
- Play (pentru acest lucru puteti folosi librarii 3rd party)

INPUT:
Add_song - calea catre o melodie (fisier mp3/wav/etc) + metadate => va returna un ID din DB
Delete_song - dupa ID-ul din baza de date
Modify_data - pentru un anumit ID se vor putea specifica valori noi pentru anumite campuri
Search - criterii de cautare (formatul va fi definit de dezvoltator) (ex: artist=Queen si format melodie=mp3)
Create_save_list - cale arhiva de output si criteriile de cautare(ex: artist=Queen si format melodie=mp3)

OUTPUT:
Rezultatele functiilor apelate
Loguri rulare program
Erori ce pot aparea

Name: FindMissingBytes	**ID:10**	**Difficulty: A**	**Propose: NIP**

Sa se scrie un tool care primeste un hash pentru un fisier originial si o arhiva trunchiata (maxim “x” bytes de la finalul arhivei originale lipsesc).
Tool-ul va gasi fisierul original din arhiva (in momentul in care se va despacheta fisierul din arhiva va avea acelasi hash ca cel primit la input)
Pentru rezolvarea acestei probleme este imperativ folosirea oricarei forme de paralelizare (

multi threding/multiuprocessing / multi system / etc)
INPUT: Arhiva truncata Numele fisierului din arhiva Hash-ul expected al fisierului Optional ar fi utila si o optiune de trunchiere a unei arhive si generare a datelor de input de mai sus.
OUTPUT: Continutul fisierului dupa ce a fost dezarhivat cu success

Name: WorkerPool	**ID:11**	**Difficulty: A**	**Propose: MKY**
Dezvoltati un asamblu de descarcare pagini web ce va fi format dintr-un script master (ce va programa primele 500 de pagini din fiecare tara conform alexa : https://www.alexa.com/topsites/countries) si un script worker (script ce va descarca continutul paginilor. Master.py va scrie intr-o coada (redis sau rabbitmq) informaţiile despre paginile ce trebuiesc descarate iar worker.py (pot fi mai multe instanţe) va prelua din acea coada informaţiile şi va face descarcarea. Pentru fiecare pagina ce trebuie descarcata master.py va stoca in coada un json cu urmatoarele informatii: Link LocatieDisk (folderul in care se va salva link-ul)			
INPUT: Coada de redis/rabbit			
OUTPUT: Fisierele descarcate in locatiile specificate Logurile programelor worker.py si master.py precum si erorile aparute			

Name: EncryptedDatabase	**ID:12**	**Difficulty: A**	**Propose: MKY**
Sa se creeze un tool care sa permita criptarea anumitor fisiere intr-o baza de date. Fisierele vor fi stocate pe disk criptate intr-o anumita locatie iar medatadatele despre fisier / metoda de criptare samd vor fi tinute intr-o baza de date. Tool-ul va permite adaugare unui fisier in baza de date, citirea unui fisier din baza de date precum si stergerea unui fisier din baza de date Se accepta doar solutii ce se vor folosi de criptare asimetrica. Nu aveti voie sa folositi tool-uri deja existente.			
INPUT: Fisierul ce trebuie adaugat / citit / sters			

OUTPUT: Output-ul comenzilor rulate. Atentie! In momentul in care se citeste un fisier din baza de date criptata - tool-ul va afisa continutul decriptat

Name: Redis GUI Client	**ID: 13**	**Difficulty: A**	**Propose: ZAR**
Implementaţi un client GUI care să implementeze RESP - REdis Serialization Protocol (Detalii aici: https://redis.io/topics/protocol) Aplicaţia ar trebui să permită operaţii de bază CRUD pentru Lists, Sets, Hashes, Sorted Sets şi Strings (Detalii aici: https://redis.io/commands) Pentru partea de implementare/prezentare se poate folosi un Server Redis local. Pentru partea de GUI se poate folosi tkinter sau alte biblioteci asemănătoare. Obs: A nu se folosi biblioteci care deja implementează protocolul RESP			
INPUT: Adresa IP şi portul serverului de Redis.			
OUTPUT: Rezultatul Operatiilor			

Name: GUI Resource Monitor	**ID: 14**	**Difficulty: A**	**Propose: ZAR**
Implementaţi o aplicaţie GUI care monitorizează anumite resurse ale sistemului (CPU Usage, Memory Usage, Disk Space, Networking Usage, etc) Aplicaţia afişează în timp real grafice şi statistici pentru resursele respective. Aceste statistici vor fi salvate sub forma unui istoric, iar userul va putea vizualiza datele din anumite intervale de timp din trecut. Totodată, graficele respective vor putea fi descărcate în format jpeg/pdf. Pentru partea de GUI se poate folosi tkinter sau alte biblioteci asemănătoare.			
INPUT:			
OUTPUT:			

Name: GUI HTTP Sniffer	**ID: 15**	**Difficulty: A**	**Propose: ZAR**
Implementaţi o aplicaţie care implementează un sniffer de pachete HTTP. Aplicaţia ar trebui să permită vizualizarea real-time a requesturilor, aplicarea de filtre pe traficul de pachete (ex: requesturi venite de la o anumită adresă, requesturi de anumite tipuri: GET/POST/DELETE s.a.m.d). Totodată, pentru un anume request, ar trebuie ca			

userul să poată afla detalii despre requestul respectiv: headers, request mode, payload, etc Nu este necesar un GUI (datele se pot afişa şi în consola).Trebuie sa fie totuşi o reprezentare clară a acestor date (sa se inteleaga ce anume reprezinta). Traficul se capturează cu ajutorul bibliotecii socket iar decodarea pachetelor se va face cu struct/ctypes
INPUT:
OUTPUT:

Name: CHESS	**ID: 16**	**Difficulty: A**	**Propose: RMC**
Se va crea o interfata grafica minimala ce va oferi utilizatorului posibilitatea sa joace o partida de sah, atat cu calculatorul, cat si cu un alt oponent. Calculatorul va realiza mutari corecte, insa aleatoriu alese.			
INPUT: chess.py <tip adversar>			
OUTPUT: Interfata grafica cu tabla de sah. Dupa terminarea jocului se va afisa un mesaj corespunzator.			

Name: GAME OF MANCALA/OWARE	**ID: 17**	**Difficulty: A**	**Propose: GDT**
Se va crea o interfata grafica ce va oferi utilizatorului posibilitatea sa joace o partidă de Mancala/Oware, atat cu calculatorul, cât şi cu un alt opponent. Calculatorul va lua decizii aleatorii, însă conform cu regulile jocului. https://ro.wikipedia.org/wiki/Mancala			
INPUT: mancala.py <tip adversar>			
OUTPUT: Interfata grafica cu tabla de mancala. După terminarea jocului se va afişa un mesaj corespunzător.			

Name: GAME OF GO	**ID: 18**	**Difficulty: A**	**Propose: RMC**
Se va crea o interfata grafica minimala ce va oferi utilizatorului posibilitatea sa joace o partida de GO, atat cu calculatorul, cat si cu un alt oponent. Calculatorul va lua decizii aleatorii, insa conform cu regulile jocului.			
INPUT: checkers.py <tip adversar>			

OUTPUT: Interfata grafica cu tabla de go. Dupa terminarea jocului se va afisa un mesaj corespunzator.

Name: SNAKE	**ID: 19**	**Difficulty: A**	**Propose: ALM**
Se va crea o interfață grafică minimală ce va oferi utilizatorului posibilitatea de a juca Snake. Când script-ul se rulează, începe o sesiune de joc. O sesiune de joc va fi formată din mai multe partide de joc. După fiecare partidă de joc, se va afișa scorul obținut, cu posibilitatea de continuare sau încheiere a sesiunii. De asemenea, se va reține și cel mai mare scor, ce se actualizează, după caz, pe parcursul sesiunii de joc. La încheierea sesiunii, recordul va fi afișat. Se vor seta atat dimensiunea tablei cat și eventuale obstacole care pot exista pe tabla (printr-un fișier JSON dat la intrare)			
INPUT: snake.py <table.json>			
OUTPUT: Interfața grafică cu fereastra unde se află șarpele. Dupa terminarea jocului, se va afișa scorul cel mai mare.			

Name: GAME OF SOLITAIRE	**ID:20**	**Difficulty: A**	**Propose: ALM**
Se va crea o interfață grafică minimală ce va oferi utilizatorului posibilitatea de a juca Solitaire.			
INPUT: solitaire.py			
OUTPUT: Interfața grafică ce afișează cărțile de joc. După terminarea jocului, se va afișa un mesaj corespunzător.			

Name: STATES OF THE WORLD	**ID:21**	**Difficulty: A**	**Propose: ALM**
Se va crea un crawler ce va intra pe pagina de *Wikipedia* cu țările lumii și va reține într-o bază de date informații precum: nume, nume capitala, populatie, densitate, suprafață, vecini, limba vorbită, fusul orar, tip de regim politic (democratic, monarhie, etc)			

De asemenea, se va construi un API peste baza de date, care va avea multiple rute, apelate cu metoda GET. Aceste apeluri vor returna topul primelor 10 ţări cu : cea mai mare populaţie, cea mai mare densitate. Ex. Vreau topul primelor 10 ţări după populaţie - apelez GET pe ruta */top-10-tari-populaţie* şi voi primi un răspuns, pe care îl voi afişa pe ecran, lista aferentă Deasemenea, ar trebui sa se poată cere şi alte informaţii (de genu - toate ţările de pe GMT+2, sau toate ţările în care se vorbeşte ENGLEZA, sau toate ţările care se bazează pe un anumit tip de regim politic), la fel sub forma de rute.
INPUT: wikipedia_api.py Client.py (care sa apeleze API-ul)
OUTPUT: r = requests.get(api_url + route) print(r.text) ``` topul ţărilor cu cea mai mare densitate```

Name: Browser Total Commander	**ID:22**	**Difficulty: A**	**Propose: GDT**
Realizati o pagina web care sa aiba backend-ul scris complet în Python, şi care sa mimeze comportamentul unui tool similar cu Total Commander. Backend-ul va fi rulat pe masina curentă iar site-ul va fi accesat pe 127.0.0.1. Interfaţa web trebuie sa suporte următoarele funcţii: ● Copiere fişiere / foldere (inclusiv selecţie de fişiere/ foldere) ● Stergere fişiere / foldere (inclusiv selecţie de fişiere/ foldere) ● Mutare/Redenumire fişiere / foldere (inclusiv selecţie de fişiere/ foldere) ● Creare fisier ● Creare folder ● Editare fisiere text Interfaţa web va avea doua panel-uri în care se vizualizeaza fişiere şi sub-folderele dintr-un folder (inclusiv cu informaţii despre dimensiunea fişierelor , respectiv date-time la care au fost create). Fiecare panel permite navigarea prin folderul curent, iar operatiile intre fisiere / foldere / etc se aplica intre cele doua panel-uri.			
INPUT:			
OUTPUT:			

Name: Browser RegEdit	**ID:23**	**Difficulty: A**	**Propose: GDT**
Realizati o pagina web care sa aiba backend-ul scris complet în Python, şi care sa mimeze comportamentul unui tool similar cu utilitarul RegEdit din Windows. Backend-ul va fi rulat pe masina curentă iar site-ul va fi accesat pe 127.0.0.1. Interfaţa web trebuie sa suporte următoarele funcţii:			

- Creare valori de regiştri (string, dword, multi-string, buffer)
- Creare cheie de regiştri
- Redenumire cheie de regiştri
- Redenumire nume pentru valori ale registrilor
- Ştergere cheie / valoare dintr-un registru
- Editare valoare a unui registru (daca este de tipul string)
- Căutare a unei valori/inclusiv recursiv într-o cheie de regiştri (valori string)

Interfaţa web va avea un mod de vizualizare similar cu cel din interfata de la utilitarul RegEdit din Windows (două panel-ul - cel din stanga un arbore prin care se poate naviga prin regiştri , cel din dreapta o lista cu cheile de regiştri şi valorile lor)

INPUT:

OUTPUT:

Name: Mahjong	**ID:24**	**Difficulty: A**	**Propose: WLD**
Se va crea o interfaţă grafică ce va oferi utilizatorului posibilitatea de a juca Mahjong.			
INPUT: mahjong.py			
OUTPUT: Interfaţa grafică ce afişează cărţile de joc.			

Name: Backgammon	**ID:25**	**Difficulty: A**	**Propose: WLD**
Se va crea o interfaţă grafică ce va oferi utilizatorului posibilitatea de a juca table (fie cu un oponent uman, fie doar cu calculatorul). Calculatorul poate sa ia decizii random (insa trebuie sa execute corect mutarea).			
INPUT: backgammon.py			
OUTPUT: Interfaţa grafică ce afişează tabla de joc.			

Name: Scrabble	**ID:26**	**Difficulty: A**	**Propose: WLD**
Se va crea o interfaţă grafică ce va oferi utilizatorului posibilitatea de a juca scrabble. Dicţionarul de cuvinte va fi dat ca parametru.			
INPUT: scrabble.py <dict_file>			

OUTPUT: Interfaţa grafică ce afişează tabla de joc.

Name: SVG renderer	**ID: 27**	**Difficulty: A**	**Propose: GDT**
Realizati o aplicatie care primeste la intrare un fisiser in formatul SVG si il randeaza intr-un fisier PNG. NU aveti voie sa folositi o librarie care stie sa faca deja acest lucru, insa puteti folosi o librarie de parsare a unui XML sau librarii grafice care sa permita creare si scrierea intr-un fisier PNG. https://www.w3schools.com/graphics/svg_intro.asp Nu trebuie implementate tot suportul din SVG. Trebuie insa implementate: Rectangle, Circle, Elipse, Line, Path, Polilyne			
INPUT: svg.py image.svg			
OUTPUT: image.png			

Name: 4 in a ROW	**ID: 28**	**Difficulty: A**	**Propose: GDT**
Se va crea o interfata grafica ce va oferi utilizatorului posibilitatea sa joace o partidă de 4 in ROW, atat cu calculatorul, cât şi cu un alt opponent. Calculatorul va avea 3 nivele (slab, mediu si avansat). La initializare se va seta şi dimensiunea tablei de joc, respectiv cine începe primul (calculator sau human) dacă jucăm cu un AI. https://en.wikipedia.org/wiki/Connect_Four			
INPUT: 4inaROW.py <tip adversar> <celule_axa_x> <celule_axa_y> [firstPlayer] EX: 4inaROW.py computer 7 5 human			
OUTPUT: Interfata grafica cu tabla de 4 in a ROW. După terminarea jocului se va afişa un mesaj corespunzător.			

Name: TrapTheMouse	**ID: 29**	**Difficulty: A**	**Propose: GDT**
Se va crea o interfata grafica ce va oferi utilizatorului posibilitatea sa joace o partidă de TrapTheMouse, atat cu calculatorul, cât şi cu un alt opponent. Calculatorul va lua decizii bune conform cu regulile jocului (trebuei sa exista macar 3 nivele de dificultate legate de cum joacă calculatorul). Tabla de joc este generată de la început cu diverse obstacole. https://www.mathplayground.com/logic_trap_the_mouse.html			
INPUT: TrapTheMouse.py <tip adversar>			
OUTPUT: Interfata grafica cu tabla hexagonala similară cu cea din joc. După terminarea			

jocului se va afişa un mesaj corespunzător.

Name: BubbleBuster	**ID: 30**	**Difficulty: A**	**Propose: GDT**
Se va crea o interfata grafica ce va oferi utilizatorului posibilitatea sa joace o partidă de BubbleBuster, https://www.mathplayground.com/logic_bubble_blaster.html			
INPUT: BubbleBuster.py			
OUTPUT: Interfata grafica cu tabla hexagonala similară cu cea din joc. După terminarea jocului se va afişa un mesaj corespunzător.			