Name: Remove Duplicates	**ID:1**	**Difficulty: C**	**Propose: GDT**
Primind un folder la intrare, identificaţi toate fişierele duplicate (acelaşi conţinut) din acel folder şi afişaţi o listă utilizatorului, cerându-i să decidă pe care dintre duplicate să le şteargă. În funcţie de decizia utilizatorului, fişierele duplicate vor fi şterse sau nu.			
INPUT: remove_duplicate.py <folder>			
OUTPUT: The following files are identical: 1. <path>\Abc.txt 2. <path>\Test.123 3. <path>\file.log Please select the file you want to keep [1..3] ? ___			

Name: Wiki-Page-Info	**ID:2**	**Difficulty: C**	**Propose: CGI**
Scrieţi un script care va primi ca parametru un URL (un link către o pagina wikipedia) şi va stoca pe disk în format JSON informaţii relevante găsite în pagina precum titlu, cuvantul care apare de mai multe ori, src-urile tuturor imaginilor din pagina si imaginile salvate pe disk.			
INPUT: - URL ex: https://ro.wikipedia.org/wiki/Regnul_Fungi			
OUTPUT: Un obiect in format JSON de ex: {“title” : <title>, “Most_frequent_word”: <word>< “Images”: <list_src_img>” }			

Name: Spanzuratoarea	**ID:3**	**Difficulty: C**	**Propose: MLA**
Sa se scrie o aplicatie in care utilizatorul trebuie sa ghicesca un anumit cuvant. Cuvintele vor fi predefinite si vor avea o anumita categorie. La rulare userul alege o categorie si se va alege un cuvant random din cele existente in categoria aleasa. Utilizatorul poate incerca cate o litera odata. Daca ghiceste o litera, atunci i se vor afisa pozitiile din cuvant pentru litera respectiva. Utilizatorul are voie sa greseasca literele de un anumit numar maxim de incercari (in functie de lungimea cuvantului). In timpul jocului se va afisa numarul de incarcari ramase. La final, se va afisa cuvantul si numarul de incercari esuate. Cuvintele vor fi salvate in fisiere specifice categoriilor din care fac parte. De asemenea, se va tine evidenta scorurilor (tot intr-un fisier).			
INPUT: sport			
OUTPUT: FOTBAL, 2 (utilizatorul a incercat 2 litere care nu exista in cuvant)			

Name: Statistics	**ID:4**	**Difficulty: C**	**Propose: CGI**
Scrieti un script care va primi ca parametru un path catre un fisier cu extensia .**csv** in care vor fi salvate pe coloane urmatoarele date : varsta, sex, IQ, nationalitate si care va calcula pentru o coloana la alegere media, mediana, deviatia standard, min, max, cvartile, covarianta si coeficientul de corelatie dintre varsta si IQ si va afisa relatia dintre cele doua variabile folosind un **“plot”**.			
INPUT: Un path catre un fisier **.csv**			
OUTPUT: Statistici referitoare la datele aflate in fisier.			

Name: Password Generator	**ID:5**	**Difficulty: C**	**Propose: CGI**
Scrieti un script care va generala si va afisa pe ecran un string reprezentand o parola care va respecta urmatoarele conditii: - Va avea o lungime intre 12 si 18 caractere - Va incepe cu o majuscula - Va contine caractere alfanumerice si minim un simbol din multimea (“!” , “?”, “#”, “@”) In functie de un parametru dat la linia de comanda parola generata va fi generata din caractere random dar care respecta conditiile de mai sus sau va fi alcatuita din cuvinte dintr-un dictionar salvat pe disk intr-un fisier **.txt** .			
INPUT: generate_password.py generate_password.py -use_dict “dictionar.txt”			
OUTPUT: exemplu de parola generata fara a folosi un dictionar : K21!Lm#n exemplu de parola generata folosind dictionar : PisicaAer@1			

Name: Remove old files	**ID:6**	**Difficulty: C**	**Propose: TDY**
Creati un script care primeste de la linia de comanda un path catre un director. Acesta se va uita in director la fisiere (recursiv in tot folderul) si va lua cateva fisiere care au fost accesate cu cel mai mult timp in urma. Apoi le va indica utilizatorului, ii va spune cand au fost accesate si cat spatiu ocupa (intr-un format intuitiiv: de ex. 3.1KB, 1.5MB, 2.1GB) si il va intreba pe care doreste sa le stearga. In functie de raspuns, acesta face actiunea respectiva			
INPUT: remove_old_files.py root_folder_path			
OUTPUT:			

Name: CommonPhrases	**ID:7**	**Difficulty: C**	**Propose: GDT**
Creaţi un script care primeşte de la linia de comanda doua fişiere text xi returnează propoziţiile comune (doua propozitii sunt identice dacă au aceleaşi cuvinte în aceeaşi ordine). Totuşi tine-ti minte ca doua cuvinte pot fi separate prin mai multe spaţii. Deasemenea, nu uitaţi într-o linie pot exista mai multe propozitii.			
INPUT: common_phrases.py file1 file2			
OUTPUT: lista cu frazele comune			

Name: OwnRM	**ID:8**	**Difficulty: C**	**Propose: WLD**
Realizaţi un script care să imite funcţionalitatea utilitarului “rm” din Linux.			
INPUT: rm -rf --interactive ../*.*			
OUTPUT:			

Name: FolderToJson	**ID:9**	**Difficulty: C**	**Propose: NIP**
Creati un tool care primeste ca parametru calea catre un folder si care genereaza un fisier json cu informatii despre structura de directoare primita ca parametru. In fisierul de output vor aparea informatii despre toate fisierele si subfolderele din interior precum si statistici despre numar de fisiere, numar de fisiere cu anumite extensii, subfoldere si dimensiune totala de pe fiecare nivel.			
INPUT: Calea catre folder			

Calea catre fisierul de output Lista de extensii de fisiere care sa apara in statistici Optiunea generare.
OUTPUT: Fisierul json generat pentru optiunea de generare Formatul fisierului va fi stabilit de dezvoltator

Name: StockBuster	**ID:10**	**Difficulty: C**	**Propose: NIP**
Scrieți un tool care va prezice necesarul de stoc pentru un magazin pentru o perioada imediat următoare. Ştiind inventarul curent de produse precum şi istoricul de vanzari tool-ul va genera o lista de comenzi de produse ce ar trebui făcute pentru a asigura un stoc suficient			
INPUT: Lista de vanzari pentru produsele magazinului: produs, cantitate, timestamp Inventarul de produse Perioada de timp pentru care trebuie estimat necesarul Formatele fişierelor şi a bazei de date vor fi alese de dezvoltator			
OUTPUT: Lista de comenzi ce trebuiesc trimise pentru perioada estimată : produs, cantitate			

Name: File searcher	**ID: 11**	**Difficulty: C**	**Propose: ZAR**
Implementaţi un script care implementează funcţionalitatea comenzii “find” din linux.			
INPUT:			
OUTPUT:			

Name: WC	**ID:12**	**Difficulty: C**	**Propose: MKY**
Scrieti un tool wc.py care sa imite comportamentul tool-ului wc din linux: https://man7.org/linux/man-pages/man1/wc.1.html Se vor considera doar input-urile primite prin sys.argv			
INPUT: Tot ce primeste la input wc			
OUTPUT: Tot ce scoate la output wc			

Name: CUT	**ID:13**	**Difficulty: C**	**Propose: MKY**
Scrieti un tool cut.py care sa imite comportamentul tool-ului cut din linux: https://man7.org/linux/man-pages/man1/cut.1.html Se vor considera doar input-urile primite prin sys.argv			
INPUT: Tot ce primeste la input cut			
OUTPUT: Tot ce primeste la output cut			

Name: UNIQ	**ID:14**	**Difficulty: C**	**Propose: MKY**
Scrieti un tool uniq.py care sa imite comportamentul tool-ului uniq din linux: https://man7.org/linux/man-pages/man1/uniq.1.html Se vor considera doar input-urile primite prin sys.argv			
INPUT: Tot ce primeste la input uniq			
OUTPUT: Tot ce primeste la input uniq			

Name: File history	**ID: 15**	**Difficulty: C**	**Propose: ZAR**
Implementaţi un script al cărui rol este acela de a monitoriza o locaţie şi de a păstra un istoric al modificărilor/acceselor acesteia (structura directoare). Istoricul este alcătuit din linii stocate într-un fişier, fiecare linie având forma: Datetime: Operaţie TipItem Item - Datetime reprezintă data şi ora la care a fost făcută operaţia - Operaţie: create/access/write/delete - TipItem: tipul itemului: director sau fişier - Item: calea absolută al itemului			
INPUT:			
OUTPUT:			

Name: Watchdog	**ID:16**	**Difficulty: C**	**Propose: WLD**
Realizaţi un script care să monitorizeze starea unui proces şi să îl repornească dacă execuţia acestuia s-a oprit. Se poate configura intervalul de timp la care să fie verificată starea procesului, cât şi locaţia fişierului de log. Se va scrie în log orice modificare a stării procesului (dead / alive).			

INPUT: watchdog.py <nume_executabil> <timp_in_secunde> <nume_log> watchdog.py bitcoin_miner.exe 60 watchdog.log
OUTPUT:

Name: Car Monitorizer	**ID:17**	**Difficulty: C**	**Propose: RMC**
Se va crea un crawler pentru https://www.autovit.ro/. In fiecare zi, sunt promovate cele mai bune oferte. Sa se preia informatii despre masinile promovate pe prima pagina a site.ului, precum: nume, pret, an, kilometraj. Informatiile vor fi scrise intr-un json, ce se va mari in fiecare zi cu noi informatii. Crawler-ul trebuie sa ruleze o data pe zi.			
INPUT: autovit.py			
OUTPUT: Un fisier json in care se afla informatiile despre masinile zilei			

Name: Best Price	**ID:18**	**Difficulty: C**	**Propose: RMC**
Se va crea un crawler pentru a prelua informatii despre pretul unor produse date ca lista de input. Se va crea un fisier json in care se vor retine urmatoarele informatii: numele produsului, pretul cel mai bun, pretul cel mai mare, numarul de oferte. Site.ul ce va fi utilizat ca si suport: https://www.compari.ro.			
INPUT: compari.py <urls of products>			
OUTPUT: Un fisier json in care se afla informatiile despre preturile produselor.			

Name: COŞ CUMPĂRĂTURI CU INGREDIENTE	**ID:19**	**Difficulty: C**	**Propose: ALM**
Se va crea un crawler pentru pagina de reţete: https://jamilacuisine.ro/. Pentru fiecare reţetă, se vor prelua ingredientele şi se va crea o listă de cumpărături. Dacă pentru mai multe reţete sunt necesare aceleaşi ingrediente, se va schimba doar cantitatea. Reţetele se vor da ca şi parametru de intrare. Link-ul pentru fiecare reţetă este de forma: https://jamilacuisine.ro/{reteta}-reteta-video/ Creaţi un fişier în care să stocaţi Coşul de cumpărături obţinut la fiecare rulare a scriptului în format *json*. Dacă acesta există, reîncărcaţi-l şi continuaţi să actualizaţi peste el.			

Name: Text analyzer	ID:20	Difficulty: C	Propose: GDT
Scrieţi un script care primeşte la intrare un fişier text şi returnează următoarele informaţii: numărul de cuvinte, numărul de propoziţii, numărul de numere de telefoane CNP-uri găsite în text, numărul de numere de telefoane (se considera ca un telefon începe cu 07…) găsit în text, precum şi o statistica pe fiecare litera (case insensitive) care sa contina de cate ori apare şi în ce procent din numărul total de litere din text. Pentru CNP-uri şi Telefoane se vor calcula valorile unice (dacă un număr de telefon sau CNP apare de mai multe ori e calculat o singura data) şi se vor şi afişa pe ecran (vedeți secțiunea OUTPUT pentru un exemplu).			
INPUT: text_analyzer.py <fisier.txt>			
OUTPUT: Cuvinte = 100 Propozitii = 10 CNP(uri) = 2 [1020320123456,1020320123456] Telefoane = 3 [0740123456, 0740123457, 0740123458], Litere: A = 10 (10%) B = 2 (2%) …..			

Name: FileCryptor	ID:21	Difficulty: C	Propose: GDT
Scrieţi un script care primeşte la intrare un fişier , o comanda şi o parola (text) şi în funcţie de comanda criptează sau decripteaza fişierul primit. Criptarea se face în felul următor: pentru fiecare caracter din fişierul de intrare, se afla codul lui ascii şi se adaugă valoarea data de codul ascii al caracterului (cu indexul egal cu offsetul caracterului curent MODULO numărul de caractere din parola) din parola. Mai exact, dacă parola este “ABCD”, şi primim un fişier de 100 de octeți, primul OCTET se înlocuieşte cu valoarea lui plus 65 (codul ascii a lui ‘A’), al doilea cu valoare lui plus 66 (codul ascii a lui ‘B’), al treilea cu valoare lui plus 67 (codul ascii a lui ‘C’), al patrulea cu valoare lui plus 68 (codul ascii a lui ‘D’), al cincilea cu valoare lui plus 65 (codul ascii a lui ‘A’), al şaselea cu valoare lui plus 66 (codul ascii a lui ‘B’), s.a.m.d Dacă suma depăşeşte 255 (8 biți) se pastreaza primii 8 biti din suma. Decriptarea se face identic doar ca în loc de suma se face diferenta. Asigurati-va ca în fisierul criptat aveți un hash al fişierului inițial ca sa puteți valida că parola de decriptare este corecta.			
INPUT: FileCryptor.py crypt abc.exe my_pass ⇒ va crea fisierul abc.exe.crypted **INPUT**: FileCryptor.py decrypt abc.exe.cryped my_pass ⇒ va crea fişierul abc.exe			
OUTPUT:			

Name: Phrase to sound	ID:22	Difficulty: C	Propose: ALM

Scopul aplicaţiei este să poată converti *o propoziţie* într-un *fişier mp3*, care va stoca înregistrarea rostită a propoziţiei. Pentru asta, vor fi create două *scripturi*. *Script 1* : Se vor da ca şi parametri de intrare 2 directoare: unul în care se află mai multe fişiere text, şi unul în care se vor stoca fişierele mp3. Primul director dat la intrare se va parcurge recursiv. Fişierele vor conţine câte o propoziţie pe linie.Fiecare propoziţie va fi convertită în fişier mp3 şi stocat în directorul de output. *Script 2* : Să se creeze o interfaţă grafică minimală, în care să se poată introduce o propoziţie. Apăsând un buton de *Play*, se va converti propoziţia în sunet şi se va reda. Utilizati o librarie text2speech pentru acest lucru.
INPUT: convert_to_sound.py <director input> <director output> play_phrase.py
OUTPUT: - directorul de output populat cu fişierele mp3 aferente - interfaţa grafică minimală, care să permita redarea sunetului

Name: Countdown Timer	**ID:23**	**Difficulty: C**	**Propose: ALM**
Să se creeze o aplicaţie cu o interfaţă grafică ce reprezintă un *temporizator*. Trebuie să se poată seta un anumit număr de secunde, minute şi ore. De asemenea, când timpul setat s-a scurs, să se notifice utilizatorul atât vizual, printr-un mesaj, cât şi printr-un semnal sonor.			
INPUT: timer.py			
OUTPUT: interfaţa grafică cu timer-ul			

Name: Subnet cheat sheet	**ID:24**	**Difficulty: C**	**Propose: WLD**
Se va crea un utilitar ce va primi o clasă de IP-uri în format CIDR şi va afişa intervalul de IP-uri în format human readable. De asemenea, utilitarul poate verifica dacă un IP se află în intervalul menţionat.			
INPUT: subnet.py 172.16.0.0/12			
OUTPUT: 172.16.0.0 - 172.31.255.255			
INPUT: subnet.py 172.16.0.0/12 172.25.197.61			
OUTPUT: True			

Name: Location size getter	**ID: 25**	**Difficulty: C**	**Propose: ZAR**

Implementaţi un script care pentru o anume locaţie, returnează sizeul fişierelor ocupat pe disk. Folosiţi multi-threading pentru calculul size-ului. Scriptul ar trebui să ofere posibilitatea de a folosi filtre pe numele fişierelor. Filtrarea se poate face folosind expresii regulate.
INPUT: Get_size.py /path/to/directory --filter ".*\.exe" --filter ".*\.dll"
OUTPUT: X (X reprezintă sizeul total al fişierelor care au extensia exe şi dll)