Programarea Algoritmilor
– LABORATOR NR. 4 –

Funcții, generatori. Module. Tratarea excepțiilor

1. (Modulul `math`, funcția `math.sqrt`)
   a) Scrieți o funcție "ipotenuza" care primește ca parametri două numere *a* și *b* și care calculează valoarea lui *c* ca fiind lungimea ipotenuzei triunghiului dreptunghic cu catetele de lungime *a* și *b*.
   b) Se citește de la tastatură un număr natural *b*. Folosind funcția de mai sus, afișați în fișierul "triplete_pitagoreice.txt" pe câte o linie câte 3 numere a, b, c, astfel: b-ul este cel citit de la tastatură, a-ul este un număr natural ≤ b, iar *c* este tot un *număr natural*, ipotenuza triunghiului cu catetele *a* și *b*. ***Indicație:*** Pentru un anumit b, trebuie găsite *toate* valorile posibile pentru a și c care respectă condițiile de mai sus.
   Exemple de triplete pitagoreice (pt b-uri diferite): (3,4,5), (5,12,13), (8,15,17) etc.

2. (Modulul `math`, constanta `math.pi`)
   a) Scrieți o funcție "lungime_arie_cerc" care primește ca parametru un număr *r* și care returnează lungimea și aria cercului având raza de lungime *r*.
   b) Se citește de la tastatură un număr natural *r*, lungimea razei unui cerc. Folosind funcția de mai sus, să se afișeze pe ecran lungimea și aria cercului de rază *r*.

3. (Tratarea excepțiilor)
   a) Să se scrie o funcție "min_max" care primește un număr *variabil* de parametri, numere *naturale*, și care returnează cel mai mic și cel mai mare număr dintre cele primite ca parametri, dacă există cel puțin un parametru și dacă toți parametrii sunt numere naturale, sau returnează `None` altfel.
   b) Dintr-un fișier "numere.txt" se citesc numere care vor fi trimise ca parametri funcției "min_max", apoi se afișează pe ecran rezultatul funcției, nr_min și nr_max. În fișierul "impartire.txt" se afișează rezultatul împărțirii lui nr_max la nr_min.
   Să se trateze excepțiile care pot apărea:
   dacă nu există fișierul pentru citire, dacă nu există drept de scriere în fișierul al doilea, dacă din fișier se citește ceva care nu este număr natural, dacă împărțirea nu este una validă, etc.
   Exemplu:
   dacă "numere.txt" conține: 11 9 31 7 145 5 101 4 80
   atunci funcția "min_max" va returna (4, 145), iar în fișierul "impartire.txt" se va scrie: 36.25

4. (Generator)
   Implementați un generator infinit de numere prime. Folosind acest generator, scrieți un program care citește de la tastatură un număr natural n și afișează pe ecran:
   a) numerele prime <= n
   b) primele n numere prime

5. Scrieți un program care citește de la tastatură o listă de numere întregi, apoi trimite numerele ca parametri unei funcții "negative_pozitive" care va returna două liste, prima formată din numerele strict negative și a doua formată din numerele strict pozitive. Afișati pe ecran rezultatul funcției.

6. Fie *v* o listă formată din n numere întregi.
   Definiți complet următoarele funcții ***în cadrul unui modul*****:**
   a) citire – citește valoarea lui n și apoi cele n elemente ale listei v;

b) afișare – afișează elementele listei v pe o linie, despărțite prin câte un spațiu;
c) valpoz – construiește o listă formată din valorile pozitive din v;
d) semn – schimbă semnul fiecărui element al tabloului v.
Scrieți un program care citește lista v și afișează pe o linie valorile negative din lista v și pe o altă linie valorile pozitive din lista v, folosind apeluri utile ale funcțiilor definite anterior.

7. (Sortare cu parametrii `key`=fct_comparator și/sau `reverse`=True/False)
Din fișierul "cuvinte.txt" se citesc cuvinte care se salvează într-o listă L.
În fișierul "cuv_sortate.txt" să se afișeze, pe câte o linie, lista L sortată astfel:
a) descrescător, folosind compararea implicită
b) crescător, comparând după lungimea cuvintelor și apoi după ordinea alfabetică
c) crescător după lungimea cuvintelor, dar pentru cuvintele de aceeași lungime păstrând ordinea din lista originală.
Exemplu:
dacă L = ["zi", "ana", "sac", "acadea", "bac", "nori", "zar", "mi", "abur"]
atunci vom avea:
a) ['zi', 'zar', 'sac', 'nori', 'mi', 'bac', 'ana', 'acadea', 'abur']
b) ['mi', 'zi', 'ana', 'bac', 'sac', 'zar', 'abur', 'nori', 'acadea']
c) ['zi', 'mi', 'ana', 'sac', 'bac', 'zar', 'nori', 'abur', 'acadea']

8. Folosiți un tip de colecție care să permită memorarea numelui, vârstei și salariului unui angajat. Fișierul "angajati.txt" conține pe prima linie un număr natural nenul n și pe următoarele n linii conține informații despre câte un angajat, despărțite prin virgule. Scrieți un program care să încarce datele din fișierul text într-o listă și să afișeze următoarele informații:
a) informațiile despre un angajat al cărui nume se citește de la tastatură (pe ecran);
b) salariul maxim și numele angajaților care au salariul respectiv (pe ecran);
c) salariul mediu din firmă (pe ecran);
d) angajații sortați alfabetic după nume (în fișierul text "angajati_nume.txt");
e) angajații sortați descrescător după vârstă și apoi alfabetic după nume (în fișierul text "angajati_varsta_nume.txt");
f) angajații sortați descrescător după salariu, apoi crescător după vârstă (în fișierul text "angajati_salariu_varsta.txt").
Fiecare cerință se va rezolva în cadrul unei funcții!

9. a) Scrieți o funcție care să citească de la tastatură o listă cu elemente numere întregi. Numărul de elemente ale listei și elementele sale se vor citi în cadrul funcției.
b) Scrieți o funcție care să returneze poziția primului element mai mare decât un număr întreg x dintr-o secvență a unei liste cuprinsă între doi indici i și j ($0 \le i \le j < n$) sau valoarea -1 dacă nu există niciun astfel de element.
c) Scrieți un program care, folosind apeluri utile ale funcțiilor definite anterior, afișează mesajul "Da" în cazul în care o listă de numere întregi, citită de la tastatură, este sortată strict descrescător sau mesajul "Nu" în caz contrar.

10. a) Folosiți un tip de colecție care să permită memorarea unei entități Student, având următoarele informații: nume, grupă, numărul total de credite obținute și situația școlară a unui student (Promovat/Nepromovat). Scrieți o funcție care să determine situațiile școlare a *n* studenți ale căror date sunt memorate într-o listă t cu elemente de tip Student. Un student este considerat promovat dacă a obținut un număr de credite cel puțin egal cu un număr natural nenul c.
b) Scrieți o funcție care să sorteze elementele unei liste t, formată din *n* elemente de tip Student în ordinea crescătoare a grupelor, iar în cadrul fiecărei grupe studenții se vor ordona descrescător

în ordinea numărului total de credite obținute. (Folosiți eventual un dicționar având ca chei numerele grupelor, iar ca valori liste cu colecții care conțin restul informațiilor despre studenții din acea grupă).

11. a) Scrieți o funcție cu număr variabil de parametri care să furnizeze numărul natural obținut prin alipirea cifrelor maxime ale mai multor numere naturale nenule date (cel mult 9 numere).
De exemplu, pentru numerele 4251, 73, 8 și 133 funcția trebuie să returneze numărul 5783.
b) Scrieți o funcție cu 3 parametri nenuli de tip întreg a,b și c care să verifice dacă aceștia pot fi considerați ca fiind numere scrise în baza 2 sau nu, folosind apeluri utile ale funcției definite anterior. ***De exemplu***, pentru numerele 1001, 11 și 100 funcția trebuie să returneze valoarea 1, iar pentru numerele 1001, 17 și 100 trebuie să returneze valoarea 0.

12. a) Scrieți o funcție generică de căutare cu următorul antet: cautare(x, L, n, cmpValori)
Funcția trebuie să returneze indexul ultimei apariții a valorii x în lista L formată din n elemente, sau None dacă valoarea x nu se găsește în listă. Funcția comparator cmpValori se consideră că returnează 1 dacă valorile aflate la indecșii primiți ca parametrii sunt egale sau 0 în caz contrar.
b) Scrieți o funcție care să afișeze, folosind apeluri utile ale funcției *cautare*, mesajul DA în cazul în care o listă L formată din n numere întregi este palindrom sau mesajul NU în caz contrar.
O listă este *palindrom* dacă prin parcurgerea sa de la dreapta la stânga se obține aceeași listă.
De exemplu, lista L=[101,17,101,13,5,13,101,17,101] este palindrom.

13. Scrieți o funcție cu număr variabil de parametri care să caute un cuvânt dat în mai multe fișiere text. Funcția va scrie într-un fișier text (al cărui nume este dat ca parametru), pe câte o linie, pentru fiecare fișier de din care a citit, numele acelui fișier, apoi numerele de ordine ale liniilor pe care apare cuvântul respectiv în acel fișier.

Indicații de rezolvare:
a) Antetul funcției va fi: cautare_cuvant(cuv, nume_fis_out, *nume_fis_in)
b) Parametrul fix *cuv* reprezintă cuvântul pe care îl vom căuta, parametrul fix *nume_fis_out* reprezintă numele fișierului text în care se vor scrie rezultatele căutării, iar lista parametrilor variabili va conține numele fișierelor text în care va fi căutat cuvântul respectiv.

De exemplu, prin apelul
cautare_cuvant("lac","rez.txt", "eminescu.txt", "bacovia.txt")
se va căuta cuvântul “lac” în fișierele text “eminescu.txt” și “bacovia.txt”,
iar rezultatul căutării va fi scris în fișierul text “rez.txt”.

c) În cadrul funcției, preluăm, pe rând, numele fiecărui fișier text din lista parametrilor variabili în șirul de caractere *nfin* și efectuăm următoarele operații:
– deschidem fișierul curent folosind variabila *fin*;
– parcurgem fișierul linie cu linie și căutăm cuvântul dat pe linia curentă (variabila *linie*, de tip str);
– dacă linia curentă conține cuvântul căutat *cuv*, atunci scriem numărul de ordine *lcrt* al liniei curente în fișierul de ieșire *fout*, marcăm faptul că am găsit cuvântul în fișierul curent (folosind variabila *gasit*) și întrerupem căutarea pe linia curentă (***de ce?***);
– după ce am terminat de parcurs fișierul curent, verificăm dacă nu am găsit cuvântul căutat în el (adică testăm dacă *gasit* este 0 sau nu) și, în caz afirmativ, scriem un mesaj corespunzător;
– închidem fișierul curent.

14. Traseu
Ana locuiește în București, ea își dorește foarte mult să ajungă la prietena sa Maria ce locuiește în Brașov. În fișierul “program.txt” se găsește programul tuturor curselor din “Autogara București”.

Realizați următoarele funcții:

a) `def fisier_dict(nume_fisier)`, funcție ce primește ca parametru numele fișierului de mai sus și va întoarce o listă de dicționare. ***Exemplu:***
[{'plecare' : 'Bucuresti', 'sosire' : 'Brasov', 'ora_plecare' : '9:20', 'ora_sosire' : '13:50'},
{'plecare' : 'Sinaia', 'sosire' : 'Brasov', 'ora_plecare' : '11:20', 'ora_sosire' : '15:00'}];

b) `def timp_calatorie(ora_plecare, ora_sosire)`, funcție ce primește ca parametri ora de plecare și ora de sosire, și va întoarce timpul necesar pentru călătorie. Dacă acesta va depași ora 23:59 se va afișa un mesaj de eroare.

c) `def calatorie(lista_program)`, funcție ce primește ca parametru lista returnată de funcția fisier_dict și returnează o nouă listă de dicționare. Aceste curse sunt cele care o pot ajuta pe Ana să ajungă la destinație.

d) `def scrie_fisier(lista_dict)`, funcție ce scrie în fișierul "traseu.txt" toate cursele care o pot ajuta pe Ana să ajungă la destinație.

e) `def timp_minim(lista_dict)`, funcție ce returnează traseul cu timp minim.

Folosind apeluri utile ale funcțiilor de mai sus, realizați un algoritm care să o ajute pe Ana să ajungă la prietena sa Maria.

Exemplu:

program.txt

Bucuresti -> Brasov 10:00 17:00
Bucuresti -> Ploiesti 12:15 14:00
Ploiesti -> Comarnic 12:45 16:00
Ploiesti -> Brasov 15:00 19:30
Targoviste -> Brasov 16:00 19:00
Bucuresti -> Sinaia 16:15 19:20
Sinaia -> Brasov 20:00 22:15

traseu.txt

Plecare la ora 10:00 sosire ora 17:00 durata 7h
Plecare la ora 12:15 schimbare in Ploiesti sosire 19:30 7h 15min
Plecare la ora 16:15 schimbare in Sinaia sosire 22:15 6h

Timp minim
Bucuresti Sinaia Brasov

15. Ping

În fișierul "ping.in" se află mesajul unui ping. Realizați funcțiile:

a) `def determina_ip(nume fisier)`, funcție ce primește ca parametru numele fișierului și returnează adresa IP și DNS (numele domeniului) către care s-a dat comanda ping.

b) `def nr_pachete(nume_fisier)`, funcție ce primește ca parametru numele fișierului și returnează numărul de pachete trimise, respectiv numărul de linii de genul
64 bytes from 216.58.214.238: icmp_seq=0 ttl=56 time=16.453 ms

c) `def verificare_timp(nume_fisier)`, funcție ce primește ca parametru numele fișierului și returnează erori daca timpii prezenți în ultima linie nu sunt exact aceeași ca cei din fișier.

Folosind apeluri utile ale funcțiilor de mai sus, realizați un algoritm ce scrie în fișierul "ping.out" numărul de pachete trimise, timpul minim, maxim, mediu, IP și DNS.

Rezolvați problema folosind expresii regulate (modulul `re`)**.**

Exemplu:

ping.in

```
192-168-0-2:~ mihaela-dianabaldovin$ ping google.com -c 5
PING google.com (216.58.214.238): 56 data bytes
64 bytes from 216.58.214.238: icmp_seq=0 ttl=56 time=16.453 ms
64 bytes from 216.58.214.238: icmp_seq=1 ttl=56 time=16.136 ms
64 bytes from 216.58.214.238: icmp_seq=2 ttl=56 time=15.838 ms
64 bytes from 216.58.214.238: icmp_seq=3 ttl=56 time=16.669 ms
64 bytes from 216.58.214.238: icmp_seq=4 ttl=56 time=15.775 ms
--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 15.775/16.174/16.669/0.345 ms
```

ping.out

Timp minim: 15.775 ms
Timp maxim: 16.669 ms
Timp mediu: 16.174,5 ms
IP: 216.58.214.238
DNS: google.com